Лекция 9.

Тема: “Сеть Кохонена”

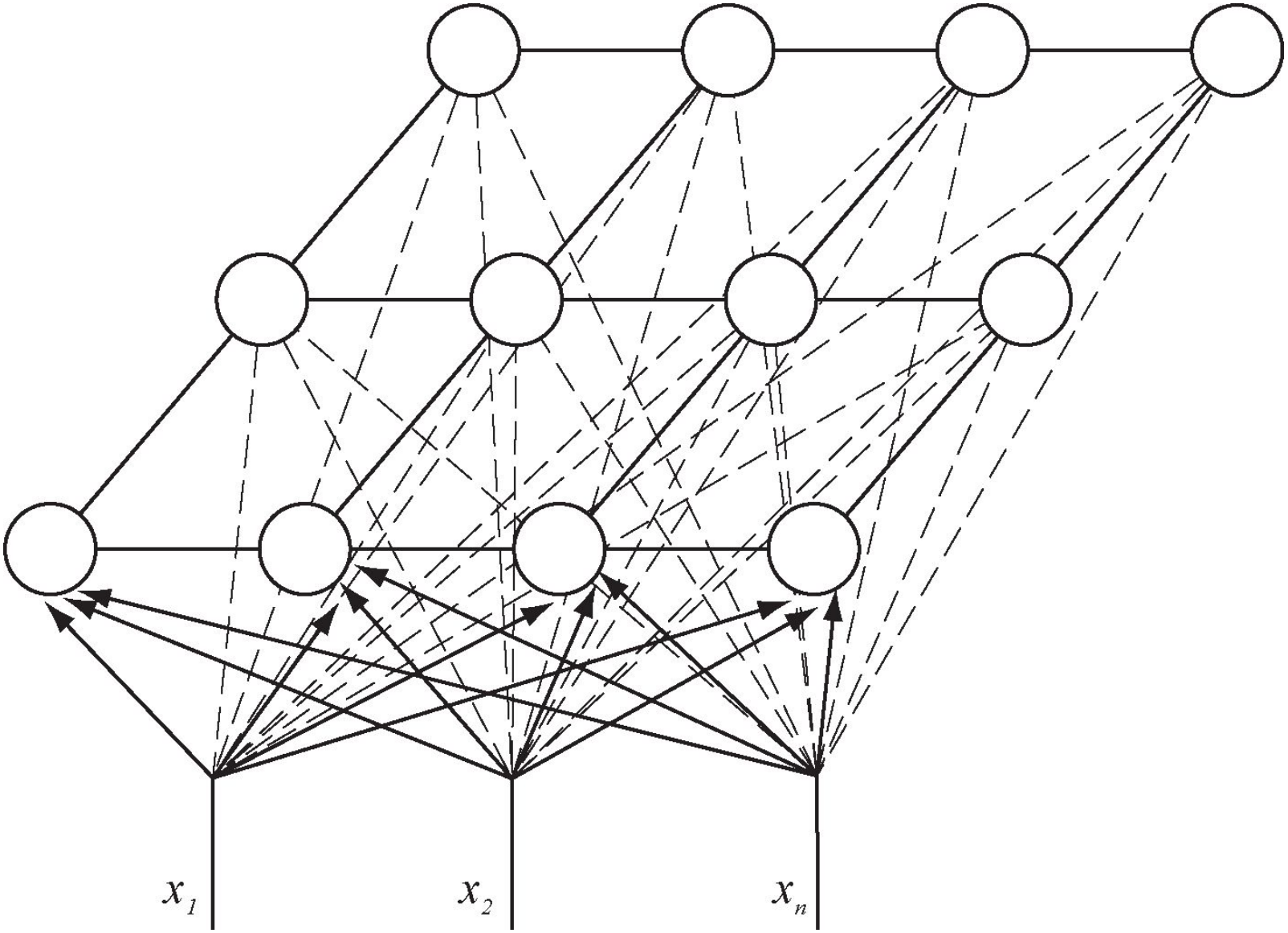

Рис.10.1 – Модель сети Кохонена

10.2 Обучение сети Кохонена

Вместо того, чтобы искать местоположение нейрона путем решения общих уравнений возбуждения, Кохонен существенно упростил решение задачи, выделяя из всех нейронов слоя лишь один с-й нейрон, для которого взвешенная сумма входных сигналов максимальна

$$c = \arg\max_j (x^T w_j). \qquad (10.1)$$

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

$$\overline{x_i} = \frac{x_i}{\|x\|}, \; i = \overline{1,N}, \qquad (10.2)$$

превращающая векторы входных сигналов в единичные с тем же направлением.

В выражении (10.2) $\|x\| = (\sum_{i=1}^{N} x_i^2)^{1/2}$.

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 (10.1) эквивалентно условию

$$c = \arg\min_j \|x - w_j\|. \qquad (10.3)$$

Вводя потенциальную функцию – функцию расстояния f_{ij} (“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 :

$$w_{ij}(k+1) = w_{ij}(k) + \alpha(k) f_{ij}(k)\left(x(k) - w_{ij}(k)\right), \qquad (10.4)$$

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

$$f_{ij}(k) = f(\|r_i - r_j\|, k) = f(d, k) = f(d, \sigma);$$

r_i, r_j - векторы, определяющие положение нейронов i и j в решетке. 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 σ, задающий величину области “соседства” и уменьшающийся с течением времени до нуля.

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

$$f_C(\|r_i - r_j\|) = f_C(0) = 1. \qquad (10.5)$$

Так как определение нейронов-победителей в сети Кохонена и в LVQ происходит идентично, то и коррекция весов этих нейронов осуществляется одинаково (см. рис.9.1).

На рис.10.2 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 – веса расположенных рядом с ним нейронов 3, 4, 6, 7. А так как нейроны 1, 2, 8, 9 лежат вне области «соседства», их весовые коэффициенты не изменяются.

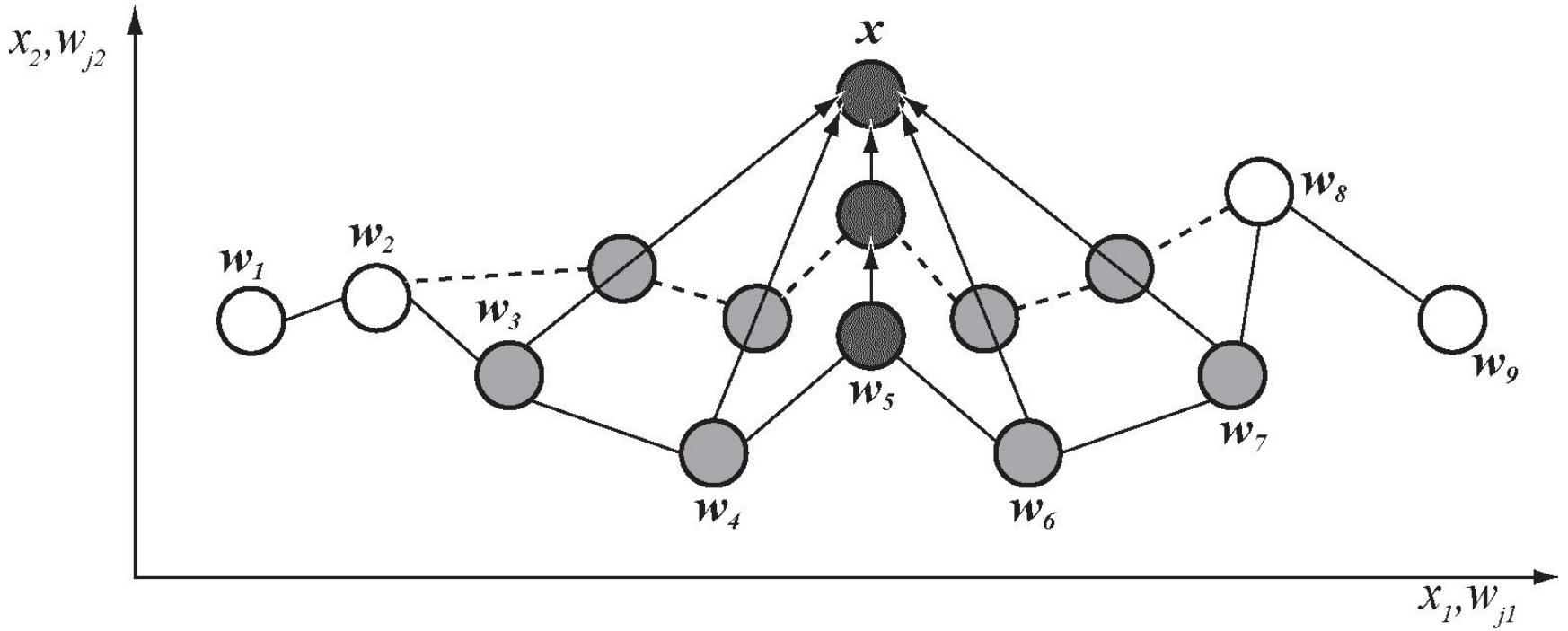

Рис.10.2 – Изменение весов карты Кохонена

Таким образом, алгоритм обучения сети Кохонена может быть описан так:

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.

2. *Выбор обучающего сигнала*

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

3. *Анализ отклика (выбор нейрона)*

По формуле (10.1) определяется активированный нейрон

4. *Процесс обучения*

Пример 10.1. Рассмотрим одномерный случай. Предположим, что сеть состоит из одного нейрона, а входной сигнал $x \in [a,b]$. Пусть начальное значение веса равно x_1 (рис. 10.3)

Рис 10.3 – Обучение сети, состоящей из одного нейрона

Каждое изменение x зависит только от удаленности от a и b т.е.:

$$\frac{dx}{dt} = \frac{\alpha}{2}[(b-x)+(a-x)] = \alpha\left(\frac{a+b}{2} - x\right),$$

где $\alpha \in (0,1]$.

Тогда перемещение x_1 в x_2 будет происходить по правилу

$$x_2 = x_1 + \gamma\left(\frac{a+b}{2} - x_1\right) = \frac{a+b}{2} + (1-\gamma)\cdot\left(x_1 - \frac{a+b}{2}\right).$$

Аналогично

$$x_3 = x_2 + \alpha\left(\frac{a+b}{2} - x_2\right) = \frac{a+b}{2} + (1-\alpha)^2 \cdot \left(x_1 - \frac{a+b}{2}\right).$$

После n-ой итерации получаем

$$x_n = \frac{a+b}{2} + (1-\alpha)^n \cdot \left(x_1 \cdot \frac{a+b}{2}\right),$$

отсюда следует, что

$$\lim_{n \to \infty} x_n = \frac{a+b}{2},$$

т.е. обучение заканчивается, когда значение веса будет делить интервал $[a,в]$ пополам. Следовательно, x_n займет устойчивое положение в середине интервала $[a,в]$.

Пример 10.2. Рассмотрим одномерный слой, состоящий из нейронов, которые делят заданный интервал $[a,b]$ на N-равных частей. Пусть веса нейронов $x_i, i=\overline{1,N}$ упорядочены, т.е. $a < x_1 < x_2 < \ldots < x_n < b$ (рис.10.4)

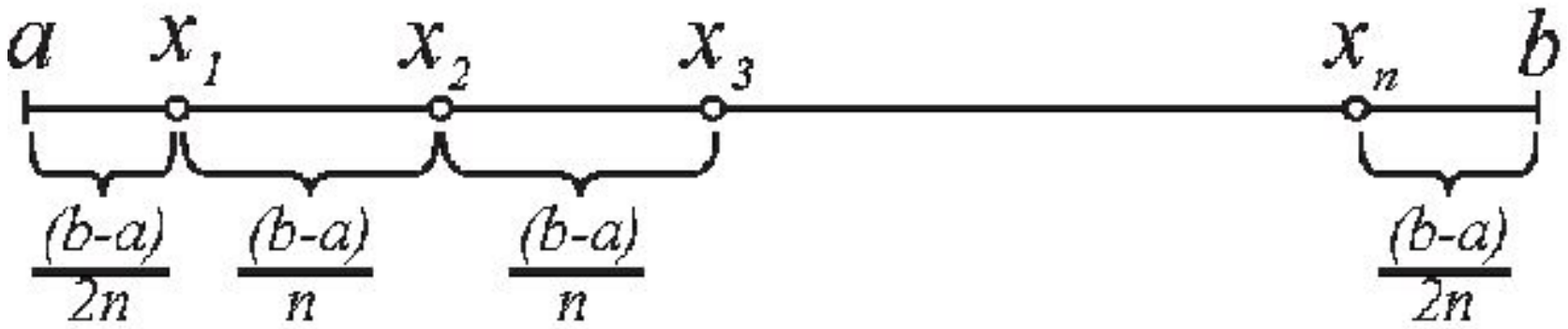

Рис 10.4 – Обучение сети

Тогда каждый вес сети может быть вычислен по формуле

$$x_i = a + (2i-1)\frac{b-a}{2N}, \qquad i=\overline{1,N}.$$

Так как эти веса делят интервал $[a,b]$ на N равных частей, то они (см. пример 10.1) занимают на этом интервале устойчивые положения.

Пример 10.3. Рассмотрим двумерный слой, состоящий из $N \times N$ нейронов. Входной вектор будет состоять из двух компонентов $x = (x_1 x_2)^T$, $x_1 \in [a,b], x_2 \in [c,d]$. Соответственно вектор весовых коэффициентов ij-нейрона имеет вид $w^{ij} = \left(w_1^{ij}, w_2^{ij}\right)^T$. Предположим, что все компоненты весового вектора упорядочены $w_1^{ij} < w_1^{ik}, j < k, w_2^{ij} < w_2^{mj}, i < m$. Находя для каждого столбца нашей матрицы весов среднее значение

$$w_1^j = \frac{1}{N}\sum_{i=1}^{N} w_1^{ij}, \qquad j = \overline{1,N},$$

с учетом упорядоченности компонентов получаем

$$a < w_1^1 < w_1^2 < \ldots < w_1^n < b.$$

Аналогично получаем для каждой строки матрицы весов

$$c_1 < w_{21}^1 < w_2^2 < \ldots < w_2^n < d.$$

По аналогии с примером 10.2 следует отметить, что в процессе обучения весовые коэффициенты разобьют интервалы $[a,b],[c,d]$ на равные части, т.е. нейроны расположатся в узлах решетки.

Пример 10.4. Пусть сеть, состоящая из трех нейронов, должна классифицировать следующие, предъявляемые ей нормализованные векторы:

$$x(1) = (0.8; 0.6)^T;\ x(2) = (0.7071; 0.7071)^T;\ x(3) = (0.6; -0.8)^T;$$

$$x(4) = (0.1961; -0.9806)^T;\ x(5) = (-0.9806; -0.1961)^T;$$

$$x(6) = (-0.9806; 0.1961)^T.$$

Случайным образом выбранные нормализованные начальные значения весов равны

$$w_1(0) = (1.000; 0.000)^T;\ w_2(0) = (0; -1)^T;\ w_3(0) = (-0.707; 0.707)^T.$$

Расположение векторов x и $w_i(0)$ показано на рис 10.5

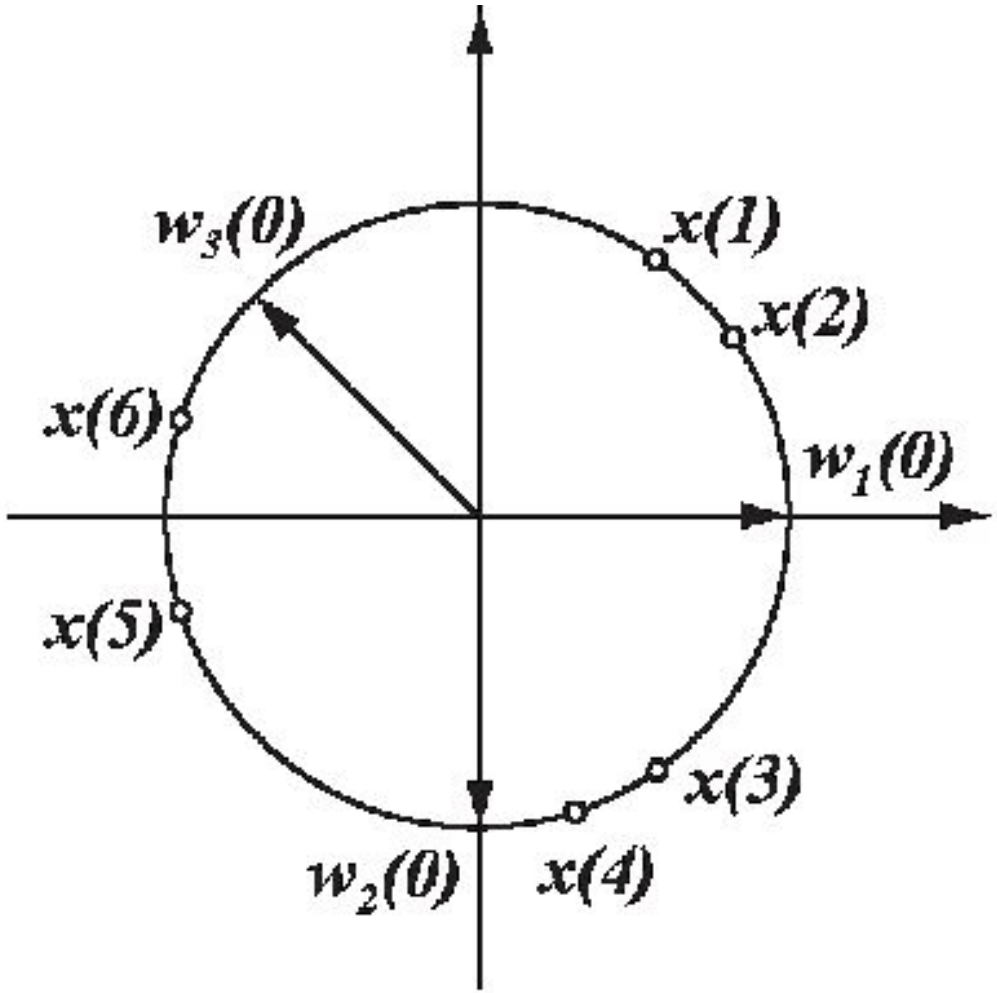

Рис.10.5 – Положение векторов x и w(0)

Предъявление сети вектора x(0) дает

$$w_1^T(0)x(1) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ -0.7 & 0.7 \end{bmatrix}^T \cdot \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.6 \\ -0.14 \end{bmatrix}.$$

Так как $\max\left[w_i^T x(1)\right] = 0.8,$ победителем будет первый нейрон. Скорректируем его веса в соответствии с алгоритмом (10.4), приняв $\alpha = 0.5$,

$$w_1(1) = w_1(0) + 0.5\left(X(1) - w_1(0)\right) = \begin{bmatrix} 1.0 \\ 0 \end{bmatrix} + 0.5\left(\begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.9 \\ 0.3 \end{bmatrix}.$$

Веса изменились таким образом, что вектор переместился к входному вектору x(1) (см. рис.10.5). Так как порядок предъявления входных векторов произволен, то при предъявлении x(4)получаем

$$w_1^T(1)X(2) = -0.11;$$

$$w_2^T(0)X(2) = -0.7;$$

$$w_3^T(0)X(2) = -0.853.$$

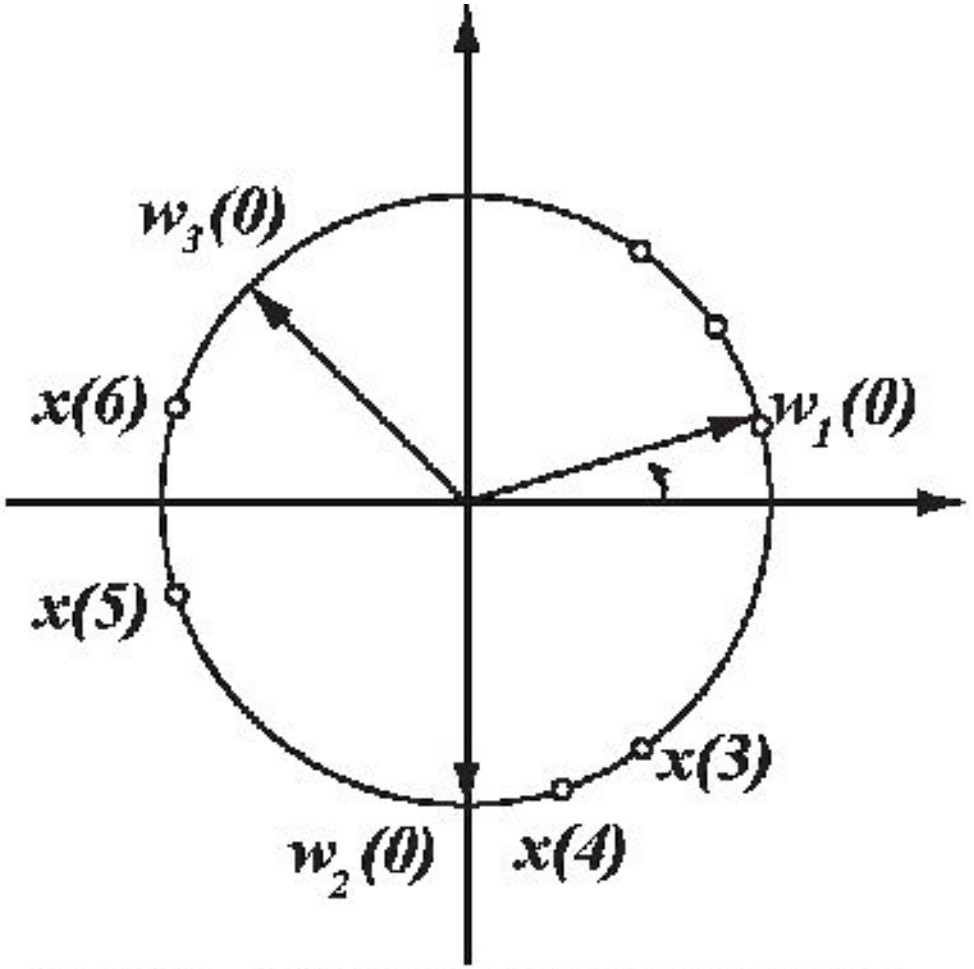

Рис 10.6 – Коррекция весового вектора w_1

Так как победителем оказался второй нейрон, скорректируем его весовые коэффициенты, меняя его положение (рис. 10.7) перемещением к x(4)

$$w_2(1) = w_2(0) + 0.5\left(X(4) - w_2(0)\right) = \begin{bmatrix} 0 \\ -1 \end{bmatrix} + 0.5\left(\begin{bmatrix} 0.1961 \\ -0.9806 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.1 \\ -0.99 \end{bmatrix}$$

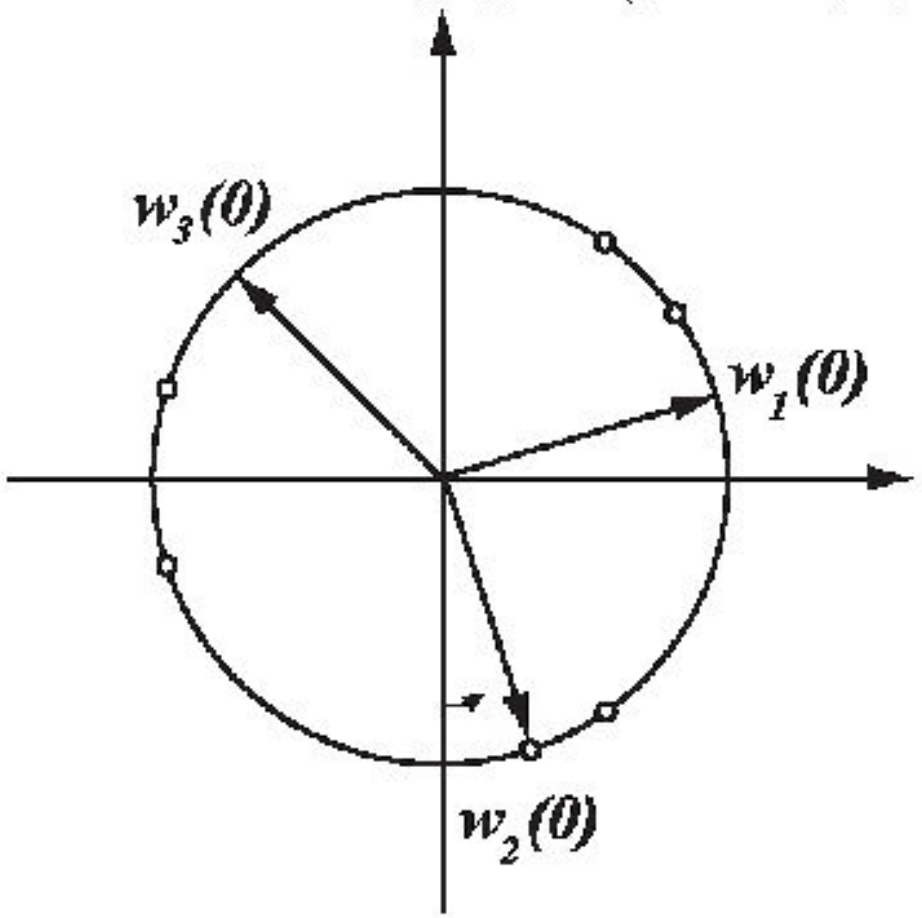

Рис 10.7 – Коррекция весового вектора w_2

Процесс обучения закончится, когда каждый весовой вектор станет прототипом различных предъявляемых сети входных сигналов, т.е. будет соответствовать своей группе входных сигналов. В конечном итоге векторы весов займут положения примерно такие же, что представлены на рис.10.8.

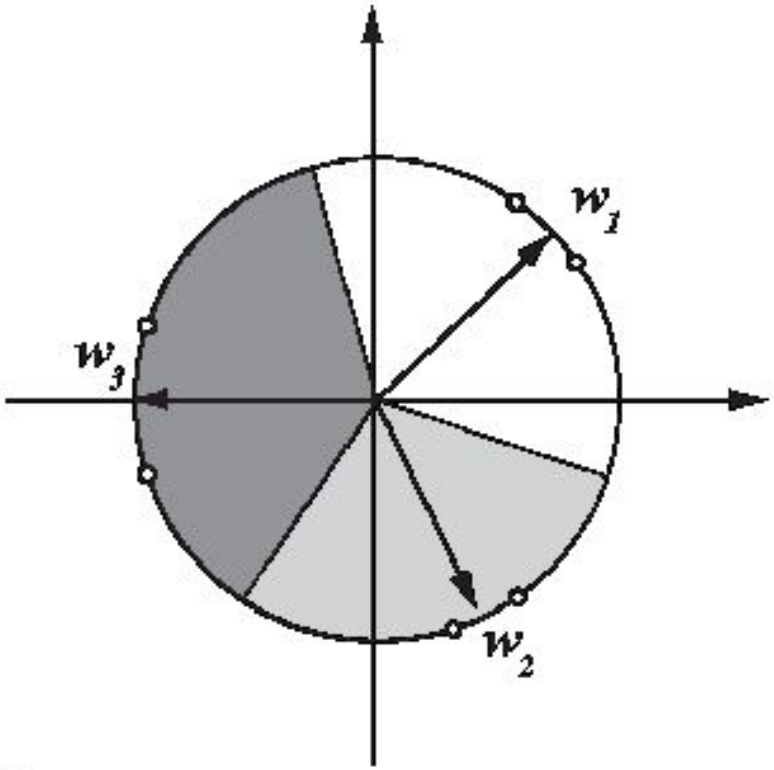

Рис 10.8– Окончательное расположение весовых векторов

Обученная таким образом сеть может классифицировать любые подаваемые на ее вход сигналы.

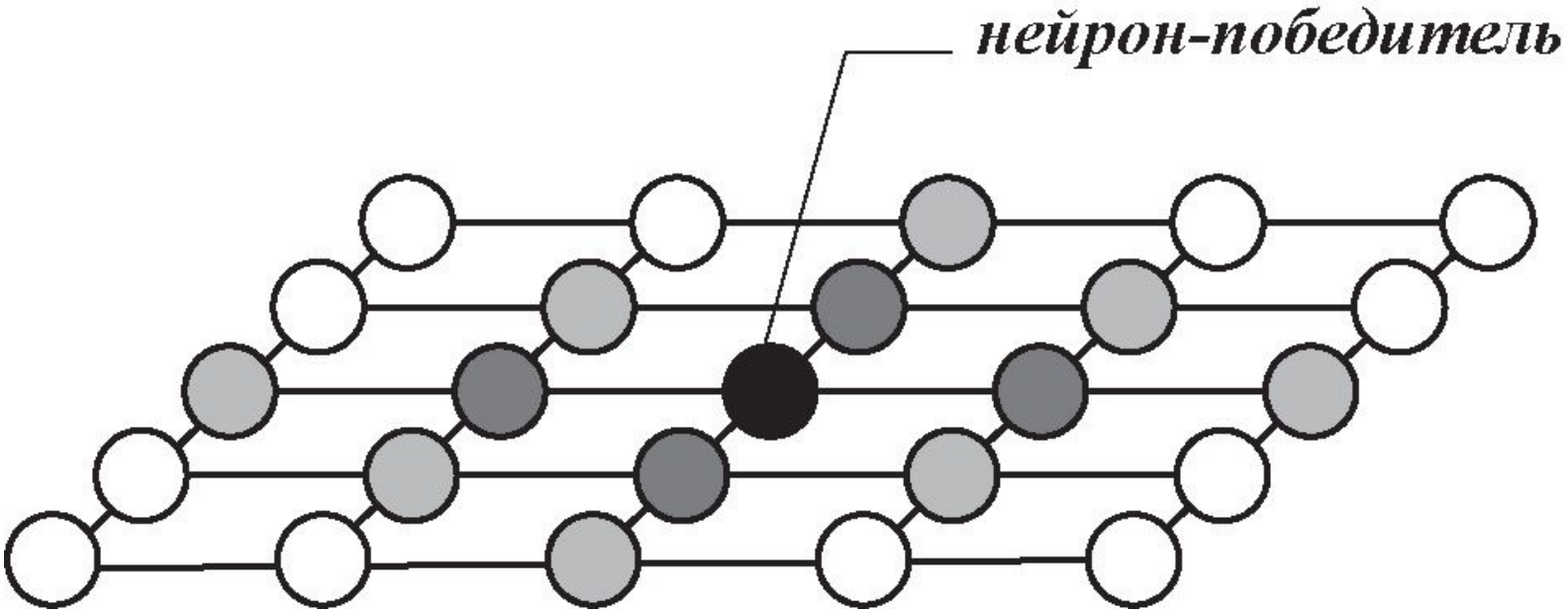

Рис 10.9 – Слой нейронов Кохонена

а) колоколообразная функция Гаусса

$$f_{gauss1}(d,\sigma) = e^{-d^2/2\sigma^2}, \qquad (10.7)$$

где σ^2 - дисперсия отклонения;

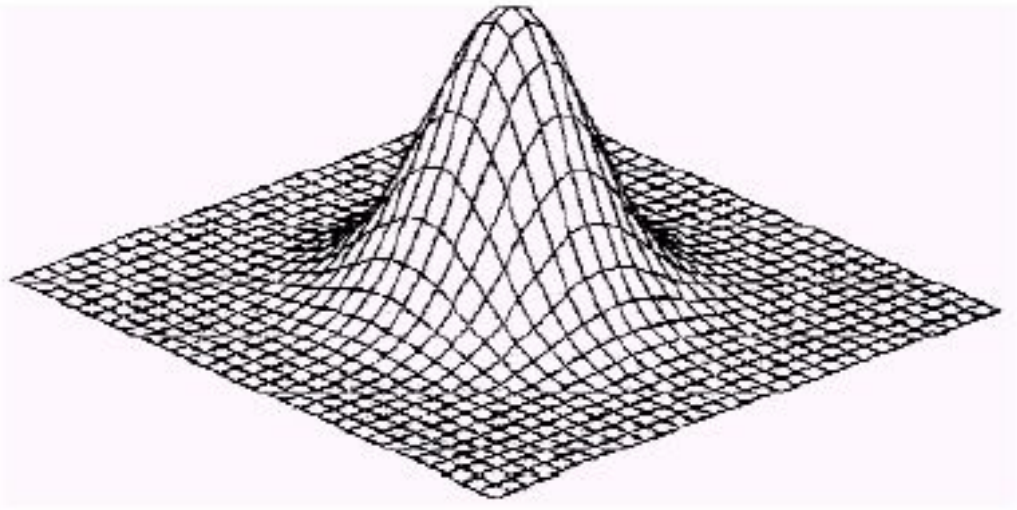

Рис.10.10 – Колоколообразная функция Гаусса

б) функция «мексиканская шляпа»

$$f_{gauss2}(d,\sigma) = (1-\left(\frac{d}{\sigma}\right)^2)\cdot e^{-\left(d/\sigma\right)^2}, \qquad (10.8)$$

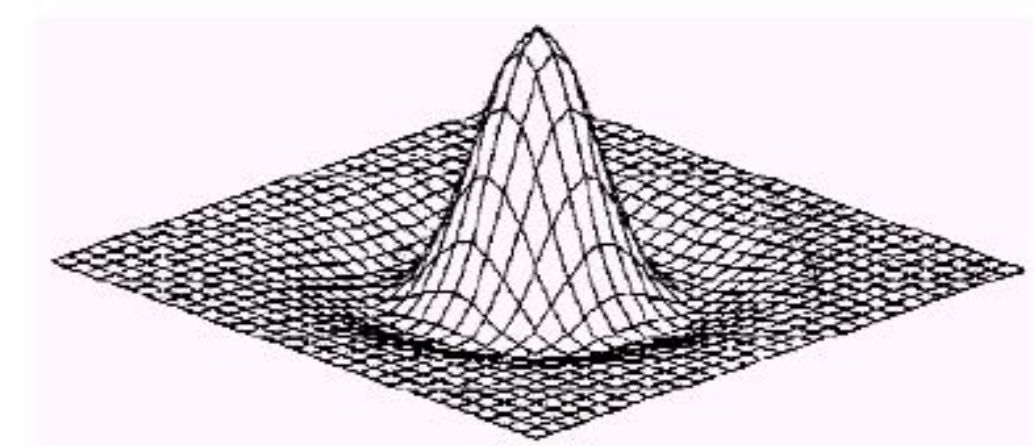

Рис.10.11 – Функция “мексиканская шляпа”

в) косинусоидная функция

$$f_{COS}(d,\sigma) = \begin{cases} \cos\left(\frac{d\pi}{2\sigma}\right), & d < \sigma; \\ 0, & d \geq \sigma, \end{cases} \qquad (10.9)$$

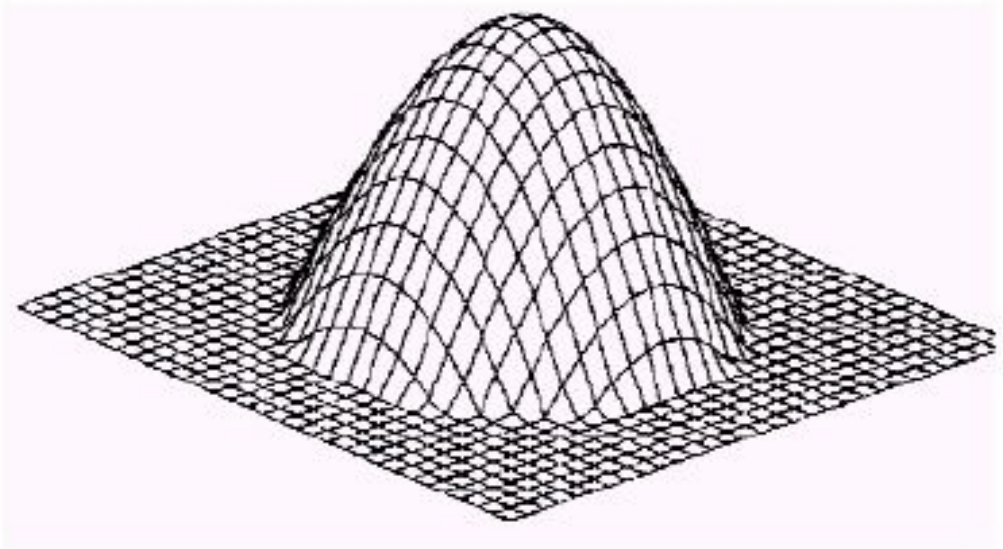

Рис.10.12 – Косинусоидная функция

г) конусообразная функция

$$f_{cone}(d,\sigma)=\begin{cases}1-\dfrac{d}{\sigma}, & d<\sigma;\\ 0, & d\geq\sigma,\end{cases} \qquad (10.10)$$

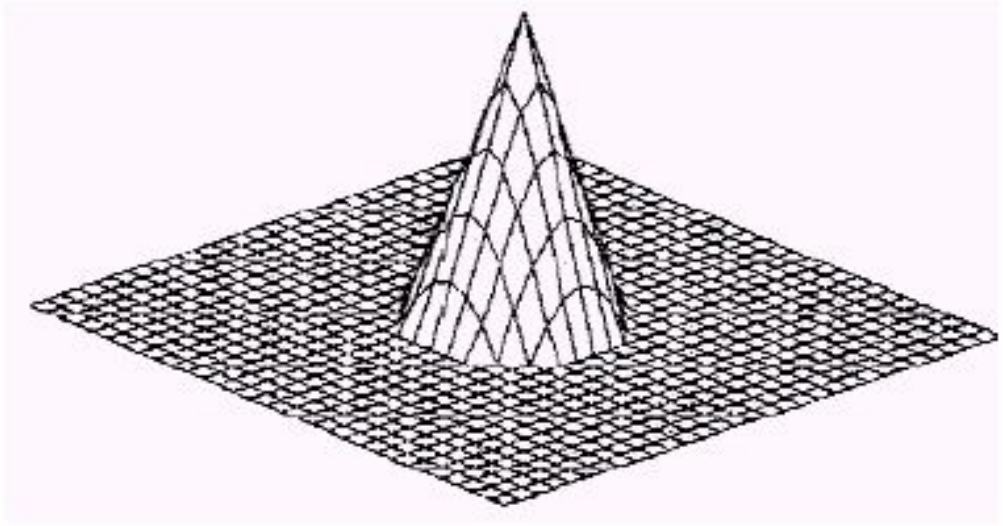

Рис.10.13 – Конусообразная функция

д) цилиндрическая функция

$$f_{cylinder}(d,\sigma) = \begin{cases} 1, & \text{при} \quad d < \sigma \\ 0, & \text{при} \quad d \geq \sigma \end{cases} \qquad (10.11)$$

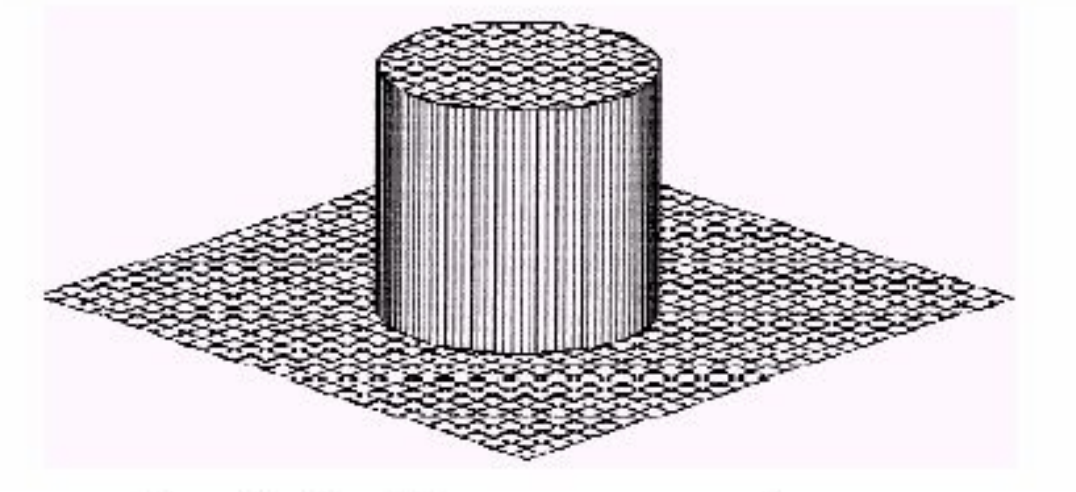

Рис. 10.14 – Цилиндрическая функция

10.4 Построение карты Кохонена

Как правило, сначала строят довольно грубую карту (модель разбиения), постепенно уточняя ее в процессе обучения. Для этого необходимо медленно изменять не только параметр α, но и, например, параметр σ в формуле (10.7). Одним из эффективных способов изменения этих параметров является следующий [ ]:

$$\alpha(k) = \alpha_0 \left[\alpha_{min} / \alpha(0) \right]^{k / k_{max}}; \qquad (10.1$$

$$\sigma(k) = \sigma(0) \left[\sigma_{min} / \sigma(0) \right]^{k / k_{max}}, \qquad (10.1$$

где $\alpha(0) \approx 0.8, \alpha_{max} << 1, \sigma(0) = 0.2, \sigma_{min} = 0.5$ - параметры, определяющие крутизну функции f_{ij} ; k_{max} - задаваемое число итераций.

На рис.10.15 показан процесс построения карты Кохонена, представляющей собой двумерную решетку, образованную путем соединения соседних нейронов, находящихся в узлах решетки. Исходя из начальных условий и используя алгоритм обучения (10.4), сеть по мере увеличения числа обучающих входных образов развивается и приобретает вид решетки. Внизу каждого рисунка поставлено количество образов, на основании которых получена соответствующая сеть [ ].

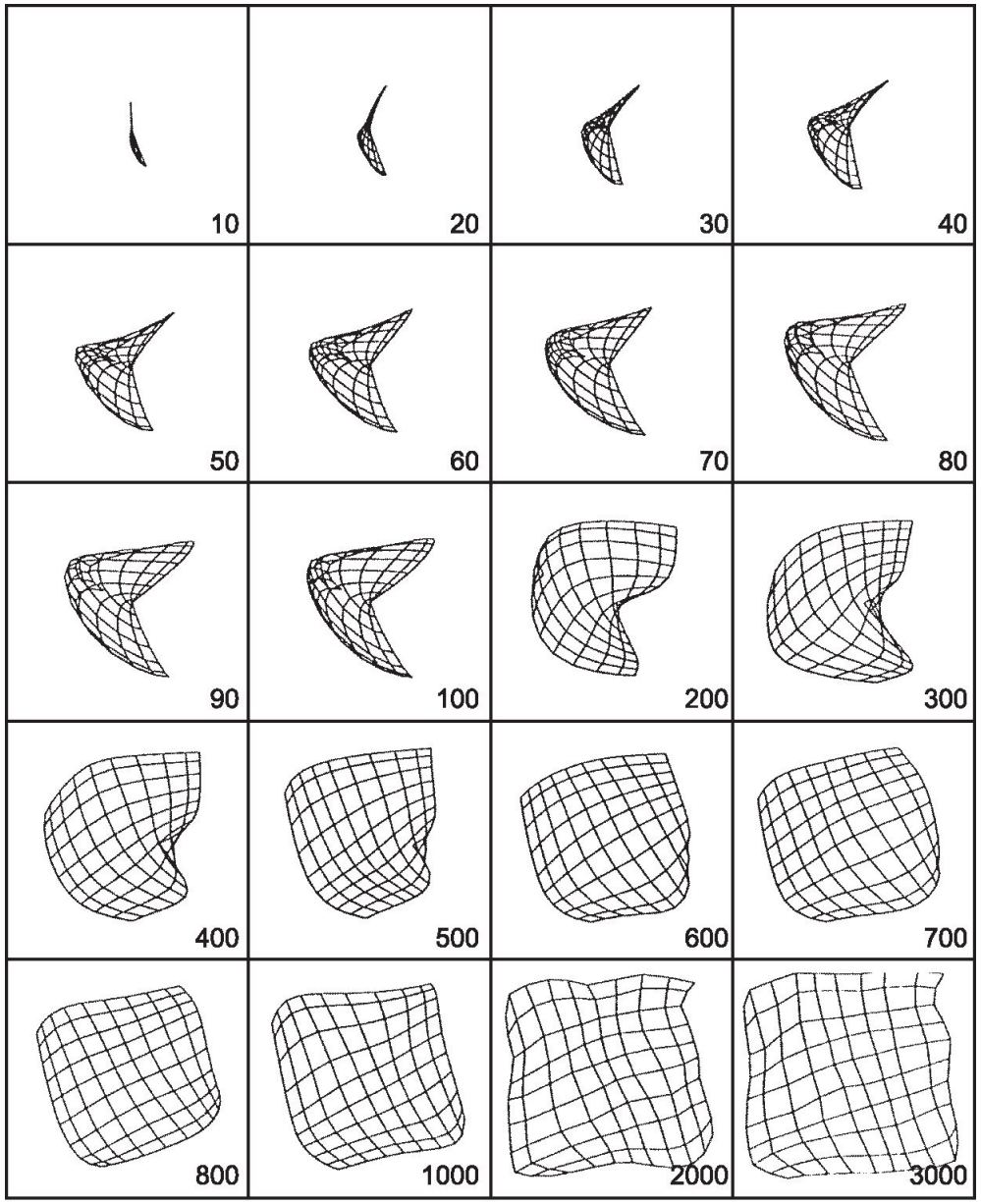

Рис.10.15 – Построение карты Кохонена

При реализации сети Кохонена возникают следующие проблемы:

1. *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 1. 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.

2. *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.

3. *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.

На рис.10.16 показан случай, когда начальное расположение весов w_1 и w_4 привело к тому, что после обучения сети все предъявляемые образы классифицируют нейроны с весами w_2, w_4, и w_5. Веса же w_1 и w_2 не изменились и могут быть удалены из сети (более подробно об этом см. раздел 19).

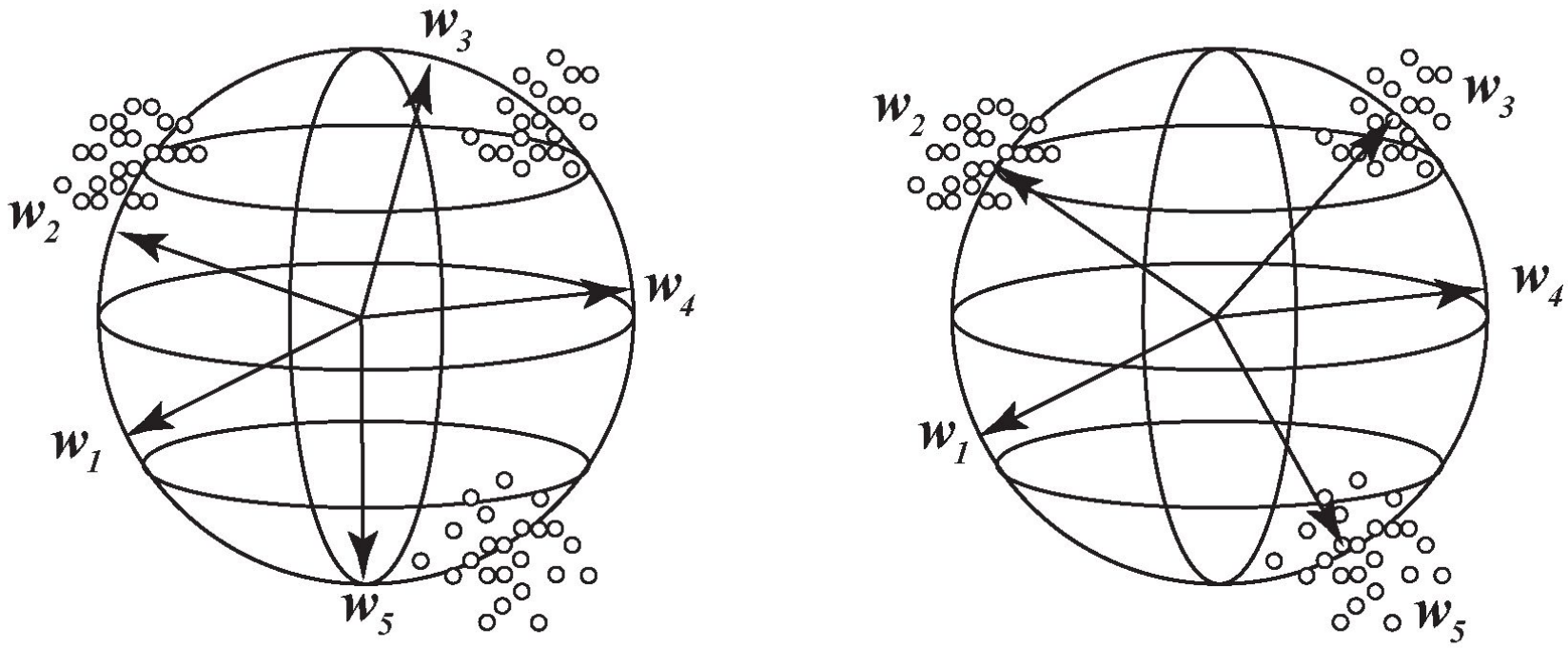

Рис.10.16 – Начальное а) и конечное б) положения векторов весов

4. *Выбор параметра расстояния* σ.

Если сначала параметр σ 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.10.17 [ ])

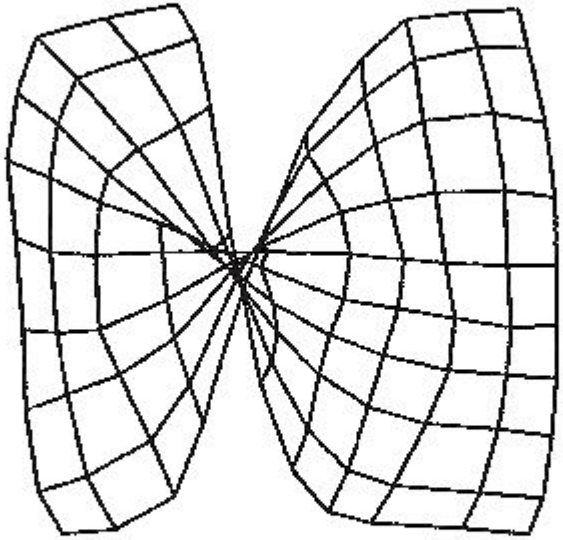

Рис.10.17 – Тополопический дефект карты Кохонена

5. *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 неизвестно.

6. *Классы входных сигналов.*

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