ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА

# Лекции по курсу «Нейронные сети и машинное обучение»

**Для студентов направления 090401**

**Очное обучение 2024-2025 учебный год**

**Лектор: Солдатова Ольга Петровна, к.т.н., доцент**

# Радиально-базисные сети. Сеть Кохонена. Рекуррентные сети. Сеть Вольтерри.

Лекция 4

**Радиально-базисные сети.**

**Алгоритм самоорганизации К-усреднений.**

**Алгоритм обратного распространения ошибки для радиально-базисной сети.**

**Гипер радиально-базисные сети.**

**Алгоритм обратного распространения ошибки для гипер радиально-базисной сети.**

**Нейрон типа WTA.**

**Сеть Кохонена.**

**Алгоритмы обучения типа WTA и WTM.**

**Гибридная сеть Кохонена.**

**Рекуррентная сеть Хопфилда.**

**Методы псевдоинверсии.**

**Рекуррентная сеть Хемминга.**

**Рекуррентная сеть Эльмана.**

**Сеть Вольтерри.**

# Литература по курсу


1. Куприянов А.В. «Искусственный интеллект и машинное обучение» https://do.ssau.ru/moodle/course/view.php?id=1459
2. Столбов В.Ю. Интеллектуальные информационные системы управления предприятием / В.Ю. Столбов, А.В. Вожаков, С.А. Федосеев. – Москва : Издательство Литрес, 2021. – 470 с. – Режим доступа: по подписке. – URL: https://www.litres.ru/book/artem-viktorovich-vo/intellektualnye-informacionnye-sistemy-upravleniya-pr-66403444/.
3. Осовский С. Нейронные сети для обработки информации / Пер. с пол. И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.: ил.
4. Горбаченко, В. И. Интеллектуальные системы: нечеткие системы и сети : учебное пособие для вузов / В. И. Горбаченко, Б. С. Ахметов, О. Ю. Кузнецова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2018. — 105 с. — (Университеты России). — ISBN 978-5-534-08359-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/424887 – Режим доступа: https://urait.ru/bcode/424887
5. 4. Гафаров Ф.М. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с. – Текст : электронный. – Режим доступа: https://repository.kpfu.ru/?p_id=187099
6. Хайкин С. Нейронные сети: Полный курс: Пер. с англ. - 2-е изд. – М.: Вильямс, 2006. – 1104 с.: ил.
7. Борисов В.В., Круглов В.В., Федулов А.С. Нечёткие модели и сети. – М.: Горячая линия–Телеком, 2007. -284 с.: ил.
8. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы: Пер. с польск. И.Д.Рудинского, - М.: Горячая линия – Телеком, 2007. – 452 с. ил.
9. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия ≪Библиотека программиста≫).
10. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А. А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.: цв. ил.

# Радиально-базисные сети

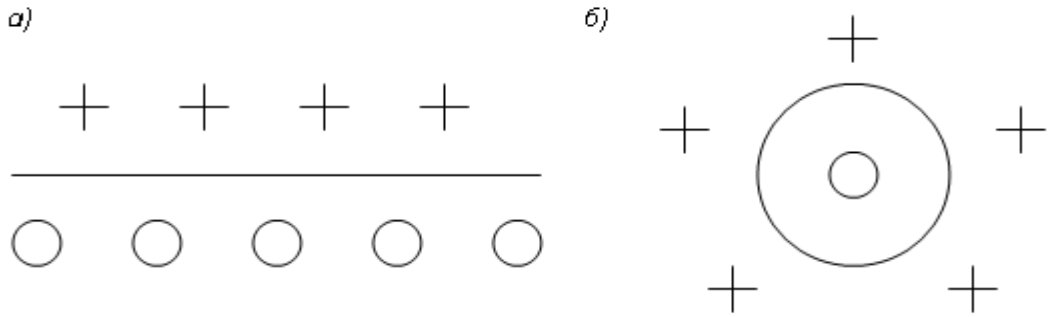
Особое семейство образуют радиальные сети, в которых скрытые нейроны реализуют функции, радиально изменяющиеся вокруг выбранного центра и принимающие ненулевые значения только в окрестности этого центра. Подобные функции, определяемые в виде $\varphi(x)=\varphi(\|x-c\|)$, называются **радиальными базисными функциями**. В таких сетях роль скрытого нейрона заключается в отображении радиального пространства вокруг одиночной заданной точки либо вокруг группы таких точек, образующих кластер. Суперпозиция сигналов, поступающих от всех скрытых нейронов, которая выполняется выходным нейроном, позволяет получить отображение всего многомерного пространства.

Сети радиального типа представляют собой естественное дополнение сигмоидальных сетей. Сигмоидальный нейрон представляется в многомерном пространстве гиперплоскостью, которая разделяет это пространство на два класса, в которых выполняется одно из двух условий:

либо $\sum_j w_{ij}x_j > 0$, либо $\sum_j w_{ij}x_j < 0$.

Такой подход продемонстрирован на рисунке 7а. В свою очередь радиальный нейрон представляет собой гиперсферу, которая осуществляет шаровое разделение пространства вокруг центральной точки (рисунок 7б).

# Радиально-базисные сети (рисунок 7)




Так как нейроны могут выполнять различные базисные функции, в радиальных сетях существует один скрытый слой нейронов. Структура типичной радиальной сети включает входной слой, на который подаются сигналы, описываемые входным вектором x, скрытый слой с нейронами радиального типа и выходной слой, состоящий, как правило, из одного или нескольких линейных нейронов. Функция выходного нейрона сводится к взвешенному суммированию сигналов, генерируемых скрытыми нейронами.

# Радиально-базисные сети

На рис. 8 представлена обобщенная структура радиально-базисной сети. В качестве радиальной функции чаще всего применяется **функция Гаусса**. При размещении ее центра в точке $c_i$ она может быть определена в сокращенной форме как:

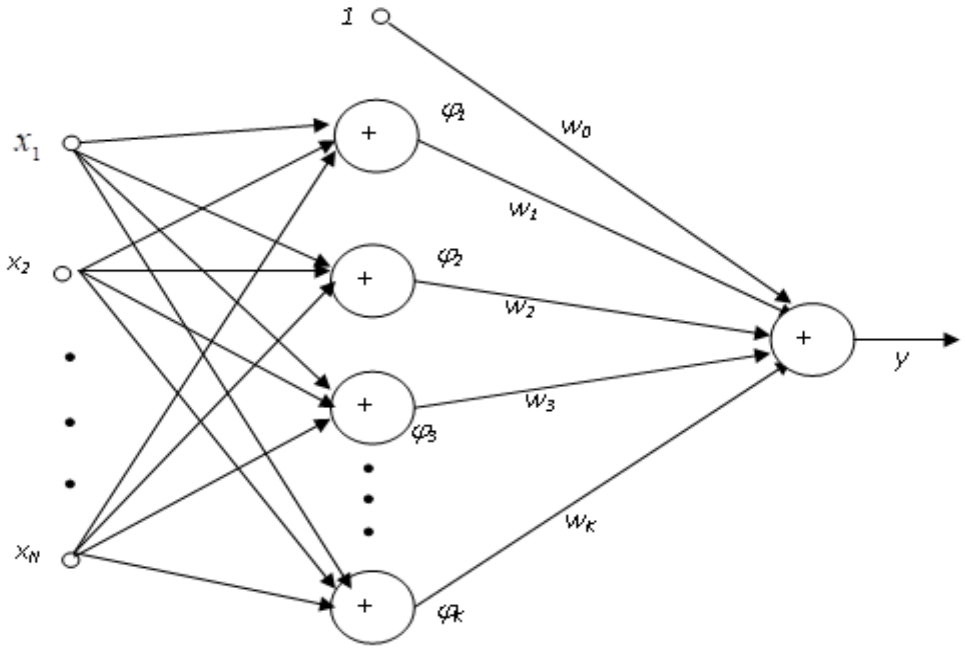
$$\varphi(x) = \varphi(\| x - c_i \|) = \exp\left(-\frac{\| x - c_i \|^2}{2\sigma_i^2}\right) \qquad (37)$$

В этом выражении $\sigma_i$ – параметр, от значения которого зависит ширина функции.

Это сеть с двухслойной структурой, в которой только скрытый слой выполняет нелинейное отображение, реализуемое нейронами с базисными радиальными функциями, параметры которых (центры $c_i$ и коэффициенты $\sigma_i$ ) уточняются в процессе обучения. Скрытый слой не содержит линейных весов, аналогичных весам сигмоидальной сети. Выходной нейрон, как правило, линеен, а его роль сводится к взвешенному суммированию сигналов, поступающих от нейронов скрытого слоя. Вес $w_0$ , как и при использовании сигмоидальных функций, представляет пороговый элемент, определяющий показатель постоянного смещения функции.

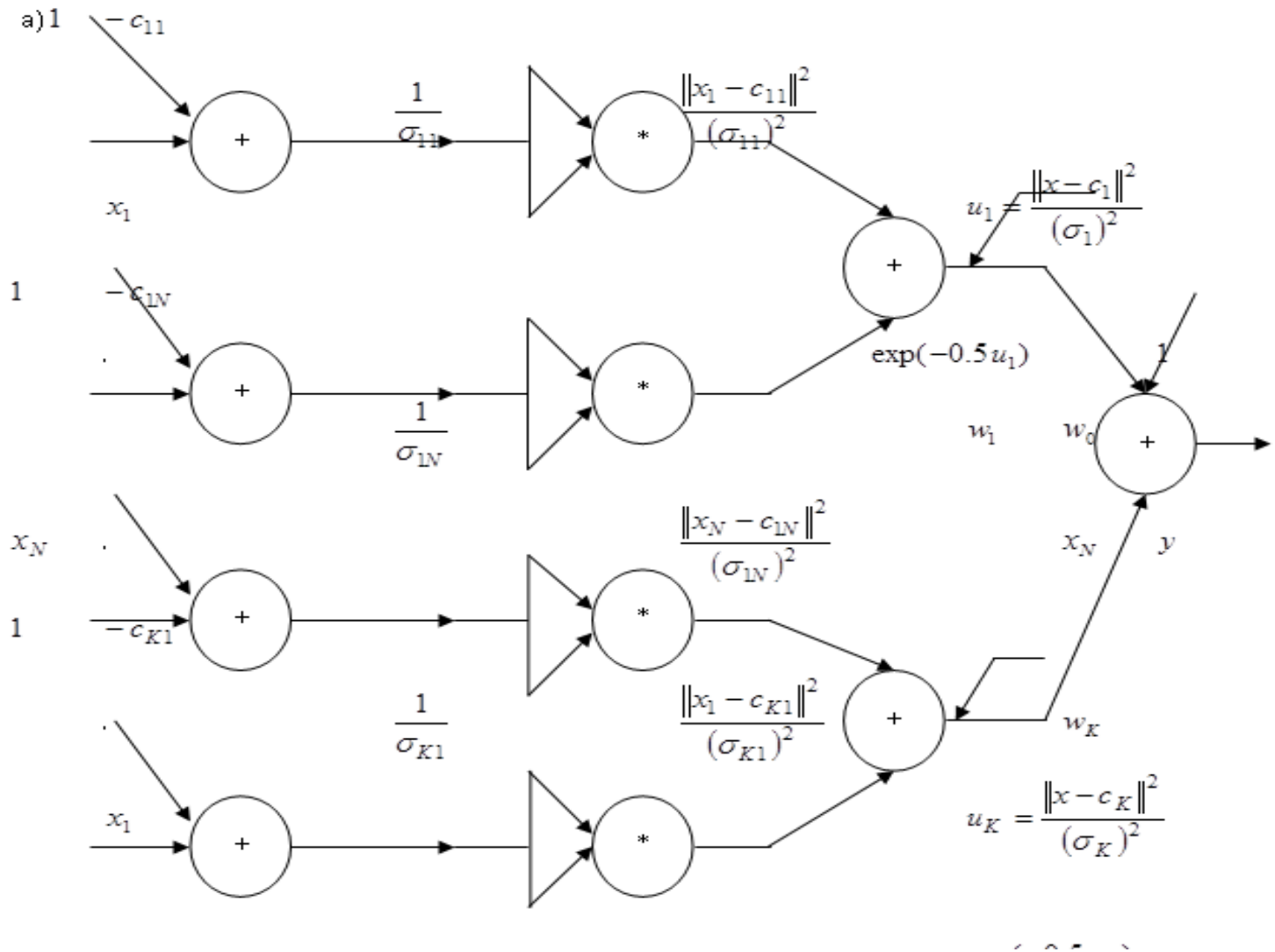
Полученная архитектура радиальных сетей аналогична структуре многослойной сети с одним скрытым слоем. Роль скрытых нейронов в ней играют базисные радиальные функции. Однако, в отличие от сигмоидальной сети, радиальная сеть имеет фиксированную структуру с одним скрытым слоем и линейными выходными нейронами. Используемые радиальные функции скрытых нейронов могут иметь разнообразную структуру.

# Радиально-базисные сети (рисунок 8)




Нелинейная радиальная функция каждого скрытого нейрона имеет свои значения параметров, тогда как в сигмоидальной сети применяются, как правило, стандартные функции активации с одним и тем же параметром. Аргументом радиальной функции является эвклидово расстояние вектора x от центра $c_i$ и $\sigma_i$, а в сигмоидальной сети это скалярное произведение векторов $w^T x$. Используемые радиальные функции скрытых нейронов могут иметь разнообразную структуру. На рисунке 9 приведена детальная структура *RBF*-сети с радиальной функцией вида (37).

# Радиально-базисные сети (рисунок 9)



# Алгоритм самоорганизации К-усреднений

Процесс обучения сети RBF с учетом выбранного типа радиальной базисной функции сводится:

- к подбору центров и параметров формы базисных функций (обычно используются алгоритмы обучения без учителя);
- к подбору весов нейронов выходного слоя (обычно используются алгоритмы обучения с учителем).

Подбор количества скрытых нейронов, считается основной проблемой, возникающей при корректном решении задачи аппроксимации. Слишком малое количество нейронов не позволяет уменьшить в достаточной степени погрешность обобщения множества обучающих данных, тогда как слишком большое их число увеличивает погрешность решения на множестве тестирующих данных. Подбор необходимого и достаточного количества нейронов зависит от многих факторов. Как правило, количество базисных функций $K$ составляет определенную долю от объема обучающих данных $p$, причем фактическая величина этой доли зависит от размерности вектора $x$ и от разброса ожидаемых значений $d_t$, соответствующих входным векторам $x_t$, для $t=1,2,\ldots,p$.

Процесс самоорганизации обучающих данных автоматически разделяет пространство на кластеры. Данные, сгруппированные внутри кластера, представляются центральной точкой, определяющей среднее значение всех его элементов. Центр кластера отождествляется с центром соответствующей радиальной функции. Разделение данных на кластеры можно выполнить с использованием алгоритма **К-усреднений**.

# Алгоритм самоорганизации К-усреднений

Согласно этому алгоритму центры радиальных базисных функций размещаются только в тех областях входного пространства, в которых имеются информативные данные. Если обучающие данные представляют непрерывную функцию, начальные значения центров в первую очередь размещают в точках, соответствующих всем максимальным и минимальным значениям функции.

Пусть $N$ - число нейронов скрытого слоя, $t$– номер итерации алгоритма. Тогда алгоритм *К-усреднений* можно описать следующим образом:

1. Инициализация. Случайным образом выбираем начальные значения центров $c_i(0)$, которые должны быть различны. При этом значения эвклидовой нормы по возможности должны быть небольшими.
2. Выборка. Выбираем вектор $x_t$ из входного пространства.
3. Определение центра-победителя. Выбираем центр $c_w$, ближайший к $x_t$, для которого выполняется соотношение:

$$w = \arg\min \| x_t - c_i(t) \|, i = 1,2,\ldots,N.$$

4. Уточнение. Центр-победитель подвергается уточнению в соответствии с формулой (38):

$$c_i(t+1) = c_i(t) + \eta(x_t - c_i(t)), \quad (38)$$

где η - коэффициент обучения, имеющий малое значение (обычно η<<1), причем уменьшающееся во времени. Остальные центры не изменяются.

5. Продолжение. Увеличиваем на единицу значение t и возвращаемся к шагу 2, пока положение центров не стабилизируется.

# Алгоритм самоорганизации К-усреднений

Также применяется разновидность алгоритма, в соответствии с которой значение центра-победителя уточняется в соответствии с формулой (39), а один или несколько ближайших к нему центров отодвигаются в противоположном направлении, и этот процесс реализуется согласно выражению

$$c_i(t+1) = c_i(t) - \eta_1(x_t - c_i(t)). \qquad (39)$$

Такая модификация алгоритма позволяет отдалить центры, расположенные близко друг к другу, что обеспечивает лучшее обследование всего пространства данных ($\eta_1 < \eta$).

После фиксации местоположения центров проводится подбор значений параметров $\sigma_i$, соответствующих конкретным базисным функциям. Параметр $\sigma_i$ радиальной функции влияет на форму функции и величину области ее охвата, в которой значение этой функции не равно нулю. Подбор $\sigma_i$ должен проводится таким образом, чтобы области охвата всех радиальных функций накрывали все пространство входных данных, причем любые две зоны могут перекрываться только в незначительной степени. При такой организации подбора значения $\sigma_i$, реализуемое радиальной сетью отображение функции будет относительно монотонным.

Для расчета может быть применен алгоритм, при котором на значение влияет расстояние между *i*-м центром и его *R* ближайшими соседями.

# Алгоритм самоорганизации К-усреднений

В этом случае значение $\sigma_i$ определяется по формуле (40):

$$\sigma_i = \sqrt{\frac{1}{R}\sum_{k=1}^{R} \| c_i - c_k \|^2} \qquad (40)$$

На практике значение R обычно лежит в интервале [3; 5].

Данный алгоритм обеспечивает только локальную оптимизацию, зависящую от начальных условий и параметров процесса обучения.

При неудачно выбранных начальных условиях, некоторые центры могут застрять в области, где количество обучающих данных ничтожно мало, либо они вообще отсутствуют. Следовательно, процесс модификации центров затормозится или остановится.

Подбор коэффициента $\eta$ тоже является такой же проблемой, как и для многослойного персептрона. Если $\eta$ имеет постоянное значение, то оно должно быть мало, чтобы обеспечить сходимость алгоритма, следовательно, увеличивается время обучения.

# Метод обратного распространения ошибки для радиально-базисных сетей

Обособленный класс алгоритмов обучения радиальных сетей составляют **градиентные алгоритмы обучения с учителем**, в которых используется алгоритм обратного распространения ошибки. Их основу составляет целевая функция, которая для одного обучающего примера имеет вид:

$$E = \frac{1}{2}\left[\sum_{i=1}^{K} w_i \varphi_i(x) - d\right]^2 \qquad (41)$$

Предположим, что применяется гауссова радиальная функция вида:

$$\varphi_i(x(t)) = \exp\left(-\frac{1}{2}u_i(t)\right) \qquad (42)$$

$$u_i(t) = \sum_{j=1}^{N} \frac{(x_j(t) - c_{ij}(t))^2}{\sigma_{ij}^2(t)} \qquad (43),$$

где $i$– индекс нейрона скрытого слоя, $j$–индекс компонента входного вектора, $t$ – индекс обучающего примера в выборке.

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в два этапа.

# Метод обратного распространения ошибки для радиально-базисных сетей

На первом этапе предъявляется обучающий пример и рассчитываются значения сигналов выходных нейронов сети и значение целевой функции, заданной выражением (41). На втором этапе минимизируется значение этой функции. Подбор значений параметров можно осуществлять, используя градиентные методы оптимизации независимо от объекта обучения – будь то вес или центр. Независимо от выбираемого метода градиентной оптимизации, необходимо, прежде всего, получить вектор градиента целевой функции относительно всех параметров сети. В результате дифференцирования этой функции получим:

$$\frac{\partial E(t)}{\partial w_0(t)} = y(t) - d(t) \tag{44}$$

$$\frac{\partial E(t)}{\partial w_i(t)} = \exp\left(-\frac{1}{2}u_i(t)\right)(y(t) - d(t)) \tag{45}$$

$$\frac{\partial E(t)}{\partial c_{ij}(t)} = (y(t) - d(t))w_i(t)\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^2(t)} \tag{46}$$

$$\frac{\partial E(t)}{\partial \sigma_{ij}(t)} = (y(t) - d(t))w_i(t)\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^3(t)} \tag{47}$$

# Метод обратного распространения ошибки для радиально-базисных сетей

Если в выходном слое содержится несколько нейронов, то формулы (44-47), соответственно примут следующий вид:

$$\frac{\partial E(t)}{\partial w_{0s}(t)} = y_s(t) - d_s(t) \tag{48}$$

$$\frac{\partial E(t)}{\partial w_{is}(t)} = \exp\left(-\frac{1}{2}u_i(t)\right)(y_s(t) - d_s(t)) \tag{49}$$

$$\frac{\partial E(t)}{\partial c_{ij}(t)} = \sum_{s=1}^{M}[(y_s(t) - d_s(t))w_{is}(t)]\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^2(t)} \tag{50}$$

$$\frac{\partial E(t)}{\partial \sigma_{ij}(t)} = \sum_{s=1}^{M}[(y_s(t) - d_s(t))w_{is}(t)]\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^3(t)} \tag{51}$$

где $s$ – номер нейрона выходного слоя.

# Метод обратного распространения ошибки для радиально-базисных сетей

При использовании метода градиентного спуска формулы для корректировки параметров радиально-базисной сети примут следующий вид:

$$w_i(t+1) = w_i(t) - \eta \frac{\partial E(t)}{\partial w_i(t)}, \quad (52)$$

$$c_{ij}(t+1) = c_{ij}(t) - \eta \frac{\partial E(t)}{\partial c_{ij}(t)}, \quad (53)$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta \frac{\partial E(t)}{\partial \sigma_{ij}(t)}. \quad (54)$$

Если в выходном слое содержится несколько нейронов, то формула (52) примет следующий вид:

$$w_{is}(t+1) = w_{is}(t) - \eta_1 \frac{\partial E(t)}{\partial w_{is}(t)} \quad (55)$$

# Гипер радиально-базисная сеть

**Структуру *RBF*-сети можно усилить путем применения масштабирования входных сигналов.** Если принять во внимание, что многомерная функция может иметь различный масштаб по каждой оси, с практической точки зрения оказывается полезным уточнить норму масштабирования путем ввода в определение эвклидовой метрики весовых коэффициентов в виде матрицы $Q$.

$$\|x\|_Q^2 = (Qx)^T (Qx) = x^T Q^T Qx \qquad (56)$$

Масштабирующая матрица при $N$-мерном векторе $x$ имеет вид:

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \dots & Q_{1N} \\ Q_{21} & Q_{22} \dots & Q_{2N} \\ \dots & \dots & \dots & \dots \\ Q_{N1} & Q_{N2} \dots & Q_{NN} \end{bmatrix} \qquad (57)$$

При обозначении произведения матриц матрицей корреляции $C$ в общем случае получим:

$$\|x\|_Q^2 = \sum_{j=1}^{N} \sum_{r=1}^{N} C_{jr} x_j x_r \qquad (58)$$

Если масштабирующая матрица $Q$ имеет диагональный вид, то получаем:

$$\|x\|_Q^2 = \sum_{j=1}^{N} C_{jj} x_j^2$$

# Гипер радиально-базисная сеть

Это означает, что норма масштабирования вектора рассчитывается согласно стандартной формуле Эвклида, с использованием индивидуальной шкалы для каждой переменной $x_j$ . При $Q=1$ взвешенная метрика Эвклида сводится к классической метрике $\|x\|_Q^2 = \|x\|^2$ .

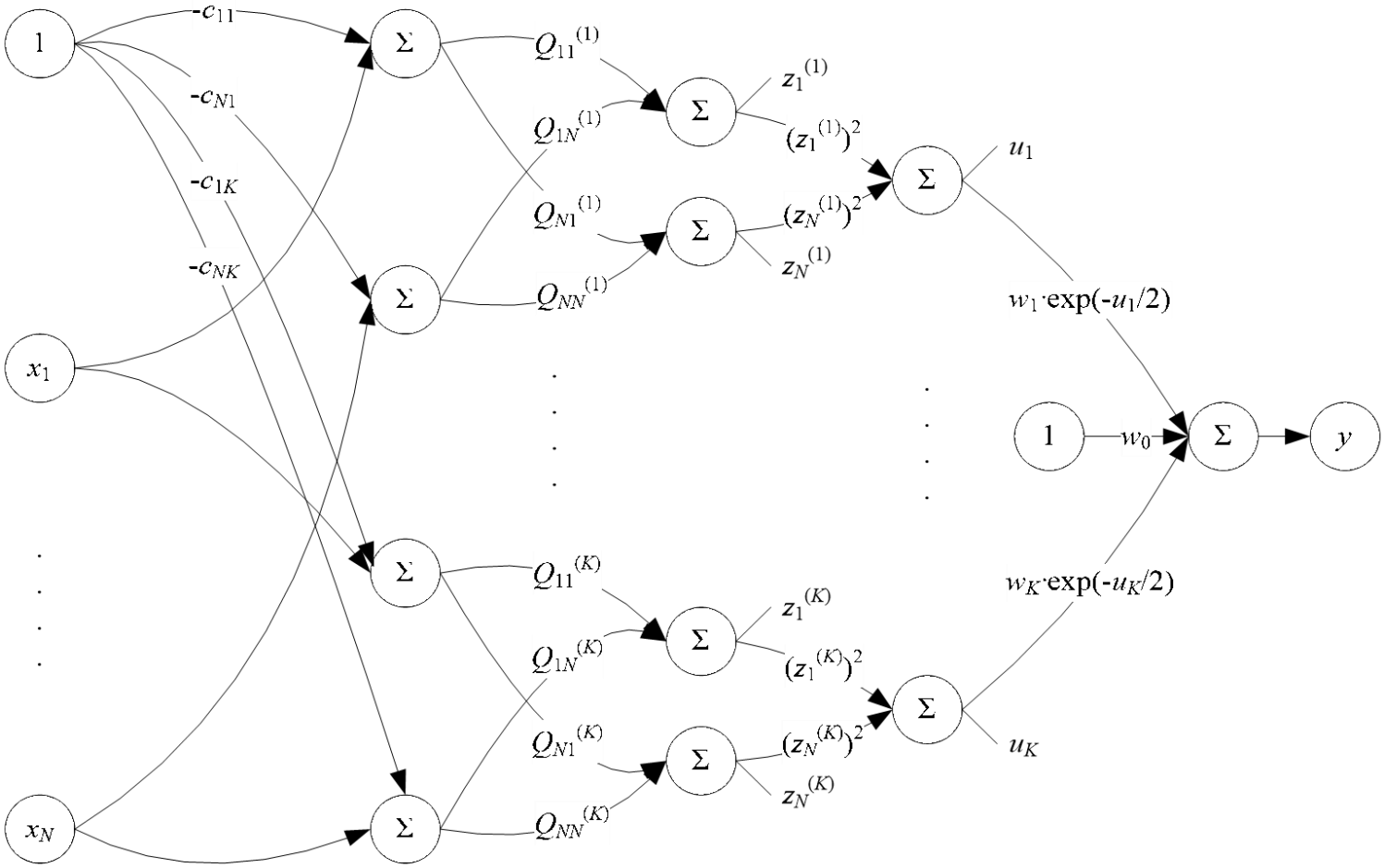
В случае использования функции Гаусса с центром в точке $c_i$ и масштабирующей взвешенной матрицы $Q_i$ , связанной с $i$-й базисной функцией, получим обобщенную форму функции Гаусса:

$$\varphi(x) = \varphi(\|x - c_i\|_{Q_i}) = \exp\left[-(x-c_i)^T Q_i^T Q_i (x-c_i)\right] = \exp\left[\frac{1}{2}(x-c_i)^T C_i (x-c_i)\right] \quad (59),$$

где матрица $\frac{1}{2}C_i = Q_i^T Q_i$ играет роль скалярного коэффициента $\frac{1}{2\sigma_i^2}$ стандартной многомерной функции Гаусса, заданной выражением (37).

Во многих практических приложениях масштабирующая матрица $Q(i)$ для $i$-го радиального нейрона имеет диагональную форму, в которой элементы главной диагонали принимают ненулевые значения. В такой системе отсутствует круговое перемешивание сигналов, соответствующих различным компонентам вектора $x$, а элемент $Q_{jj}^{(i)}$ играет роль индивидуального масштабирующего коэффициента для $j$-го компонента вектора $x$ $i$-го нейрона. В сетях ***HRBF*** роль коэффициентов $\sigma_i^2$ выполняют элементы матрицы $Q$, которые уточняются в процессе обучения (рисунок 10).

# Детальная структура радиальной сети *HRBF* с произвольной масшабирующей матрицей *Q* (рисунок 10)



# Метод обратного распространения ошибки для гипер радиально-базисных сетей

Основу градиентных алгоритмов обучения составляет целевая функция, которая для одного обучающего примера имеет вид:

$$E = \frac{1}{2}\left[\sum_{i=1}^{K} w_i \varphi_i(x) - d\right]^2 \qquad (60)$$

Предположим, что применяется **самая общая форма гауссовской радиальной функции** $\varphi_i(x)$, соответствующей сети *HRBF*, в которой

$$\boldsymbol{\varphi(x) = \exp\left[-\frac{1}{2}[Q_i(x-c_i)]^T[Q_i(x-c_i)]\right]} \qquad ,(61)$$

а матрица $Q_i$ имеет произвольную структуру.

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в два этапа. На первом этапе предъявляется обучающий вектор и рассчитываются значения сигналов выходных нейронов сети и фактическое значение целевой функции, заданной выражением (60). На втором этапе минимизируется значение этой функции.

Подбор значений параметров можно осуществлять, используя градиентные методы оптимизации независимо от объекта обучения – будь то вес или центр. Независимо от выбираемого метода градиентной оптимизации, необходимо, прежде всего, получить вектор градиента целевой функции относительно всех параметров сети.

# Метод обратного распространения ошибки для гипер радиально-базисных сетей

Аналитические выражения для частных производных можно записать в более простом виде:

$$\frac{\partial E}{\partial w_0} = y - d \tag{62}$$

$$\frac{\partial E}{\partial w_i} = \exp\left(-\frac{1}{2}u_i\right)(y-d) \tag{63}$$

$$\frac{\partial E}{\partial c_j^{(i)}} = \hat{v}_j^{(i)} = -\exp\left(-\frac{1}{2}u_i\right) w_i (y-d) \sum_{r=1}^{N} Q_{jr}^{(i)} z_r^{(i)} \tag{64}$$

$$\frac{\partial E}{\partial Q_{jr}^{(i)}} = v_t^{(i)} \hat{z}_t^{(i)} = -\exp\left(-\frac{1}{2}u_i\right) w_i (y-d)\left(x_j - c_j^{(i)}\right) z_r^{(i)} \tag{65}$$

где

$$z_r^{(i)} = \sum_{j=1}^{N} Q_{jr}^{(i)} (x_j - c_j^{(i)}) \tag{66}$$

*i*- индекс нейрона скрытого слоя, $i=1,2,\ldots,K$;

$$u_i = \sum_{j=1}^{N} \left[z_j^{(i)}\right]^2 \tag{67}$$

*j*- индекс компонента входного вектора *x*, $j=1,2,\ldots,N$;

*r*- индекс переменной в компоненте входного вектора , $r=1,2,\ldots,N$;

$Q_{jr}$ – элементы масштабирующей матрицы *Q*.

# Сравнение радиально-базисной сети и многослойного персептрона

Сети RBF и MLP являются примерами нелинейных многослойных сетей прямого распространения. И те и другие являются универсальными аппроксиматорами, однако эти два типа сетей отличаются по некоторым важным аспектам.

Сети RBF (в своей основной форме) имеют один скрытый слой, в то время как многослойный персептрон может иметь большее число скрытых слоев.

Обычно скрытые и выходные нейроны сети MLP используют одну и ту же модель нейрона. Нейроны скрытого слоя сети RBF могут отличаться друг от друга и от нейронов выходного слоя.

Скрытый слой сети RBF является нелинейным, а выходной – линейным. В то же время скрытые и выходной слой сети MLP являются нелинейными. Если сеть MLP используется для решения задач нелинейной регрессии, в качестве выходных нейронов выбираются линейные нейроны.

Аргумент функции активации каждого скрытого нейрона сети RBF представляет собой эвклидову меру между входным вектором и центром радиальной функции. Аргументом функции активации каждого скрытого нейрона сети MLP является скалярное произведение входного вектора и вектора синаптических весов данного нейрона.

Сеть MLP обеспечивает глобальную аппроксимацию нелинейного отображения. Сеть RBF создает локальную аппроксимацию нелинейного отображения.

# Нейрон типа WTA

Рассмотрим нейроны типа ***WTA*** (*winnertakesall* – победитель получает все). Эти нейроны имеют входной модуль в виде стандартного сумматора, рассчитывающего сумму входных сигналов с соответствующими весами $w_{ij}$ . Выходной сигнал *i*-го сумматора определяется согласно формуле:

$$u_i = \sum_{j=0}^{N} w_{ij} x_j \qquad (68)$$

Группа конкурирующих между собой нейронов получает одни и те же входные сигналы . Выходные сигналы нейронов сравниваются между собой, и по результатам сравнения победителем признается нейрон, значение выходного сигнала у которого оказалось наибольшим. Нейрон-победитель вырабатывает на своем выходе состояние 1, а остальные нейроны переходят в состояние 0. Для обучения нейронов типа *WTA* не требуется учитель. На начальном этапе случайным образом выбираются весовые коэффициенты каждого нейрона, нормализуемые относительно 1. После подачи первого входного вектора ***x*** определяется победитель этапа. Победивший нейрон переходит в состояние 1, что позволяет провести уточнение весов его входных линий по следующему правилу:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \dot{\eta}\left(x_j - w_{ij}(t)\right) \qquad (69)$$

# Нейрон типа WTA

Проигравшие нейроны не изменяют свои весовые коэффициенты.

Схема соединения нейронов типа WTA изображена на рисунке 11.

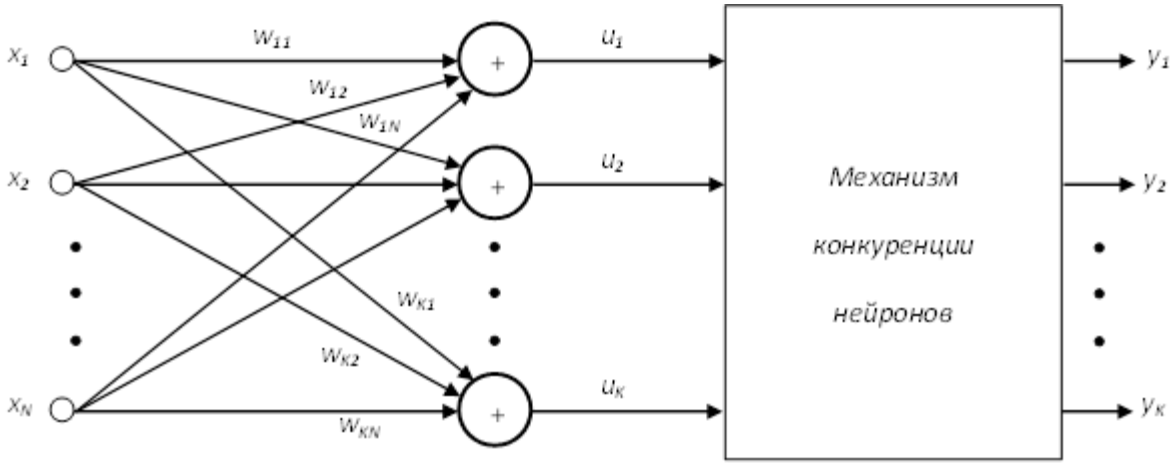
На функционирование нейронов типа WTA оказывает существенное влияние нормализация входных векторов и весовых коэффициентов. Выходной сигнал *i*-го нейрона может быть описан векторным отношением:

$$u_i = w^T x = \|w\| \|x\| \cos\varphi_i \qquad (70)$$

Поскольку $\|w\|=\|x\|=1$ , значение выходного сигнала определяется углом между векторами *x* и *w*. Поэтому победителем оказывается нейрон, вектор весов которого оказывается наиболее близким текущему обучаемому вектору.

В результате победы нейрона уточняются его весовые коэффициенты, значения которых приближаются к значениям вектора x. Проигравшие нейроны не изменяют свои веса. Следствием такой конкуренции становится самоорганизация процесса обучения.

# Нейрон типа WTA (рисунок 11)



# Сеть Кохонена

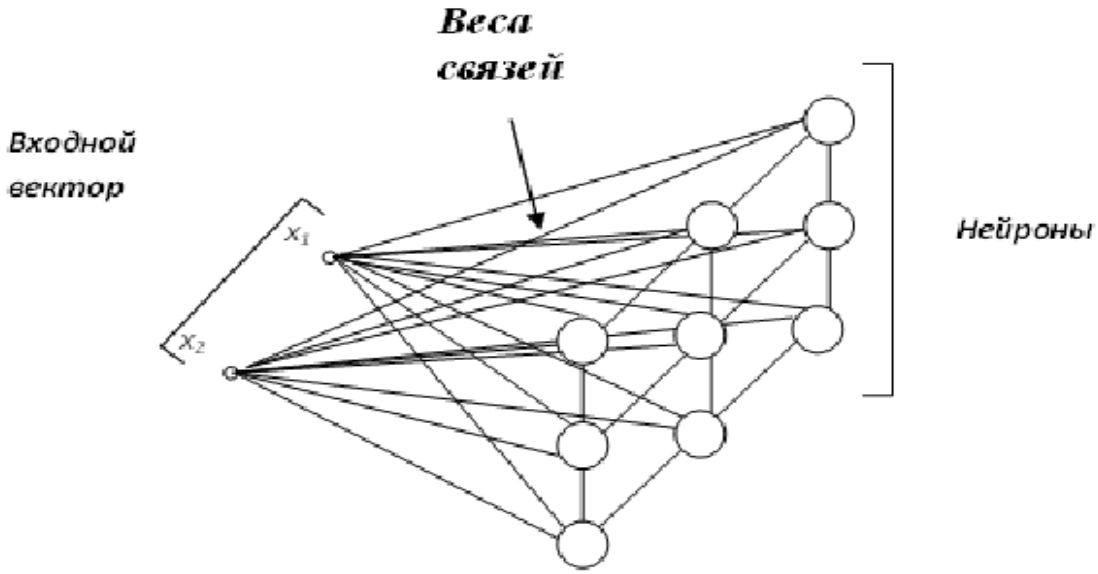
Сеть Кохонена относится к **конкурирующим сетям на основе самоорганизации**. В соответствии с поданными сигналами активным оказывается один нейрон в сети (или в группе). Выходной нейрон, который выиграл соревнование, называется **нейроном-победителем**.

Нейроны в ходе конкурентного процесса настраиваются на различные входные векторы или классы входных векторов. При этом происходит естественное расслоение нейронов на различные группы, отдельные нейроны или их группы сотрудничают между собой и активизируются в ответ на поданный сигнал, подавляя своей активностью другие нейроны. Можно говорить как о сотрудничестве между нейронами внутри группы, так и о конкуренции между нейронами внутри группы и между различными группами.

Среди механизмов самоорганизации можно выделить два основных класса: самоорганизация, основанная на **ассоциативном правиле Хебба**, и механизм конкуренции нейронов на базе **обобщенного правила Кохонена**. Сеть Кохонена реализует обобщенное правило.

Нейроны помещаются в узлах решетки, обычно одно- или двумерной. Сети более высокой размерности также возможны, но используются достаточно редко. Как правило, это однослойные сети прямого распространения, в которых нейрон соединен со всеми компонентами -мерного входного вектора x так, как это схематично изображено для  на рисунке 12.

# Сеть Кохонена (рисунок 12)




Формирование самоорганизующихся сетей начинается с инициализации синаптических весов сети. Обычно синаптическим весам присваиваются малые значения, которые формируются генератором случайных чисел. При такой инициализации сеть изначально не имеет какого-либо порядка признаков входных векторов.

# Сеть Кохонена

После инициализации сети реализуются три основных процесса:

1. ***Конкуренция.*** Для каждого входного вектора нейроны сети вычисляют относительные значения дискриминантной функции.
2. ***Кооперация.*** Победивший нейрон определяет топологическую окрестность группы нейронов, обеспечивая базис для кооперации между ними.
3. ***Синаптическая адаптация.*** Корректировка синаптических весов возбужденных нейронов позволяет увеличить их собственные значения дискриминантных функций по отношению к входным векторам. Корректировка производится таким образом, чтобы выходной сигнал нейрона-победителя усиливался при последующем применении аналогичных входных векторов.

Веса синаптических связей нейронов образуют вектор $w_i = [w_{i1}, w_{i2}, \ldots, w_{iN}]^T$ . После нормализации входных векторов при активации сети вектором $x$ в конкурентной борьбе побеждает тот нейрон, веса которого в наименьшей степени отличаются от соответствующих компонентов этого вектора. Для $w$-го нейрона-победителя выполняется соотношение

$$d(x, w_w) = \min_{1 \le i \le K} d(x, w_i) \tag{71}$$

где $d(x, w)$ обозначает расстояние между векторами $x$ и $w$, а $K$ - количество нейронов.

# Меры расстояния между векторами

Для выявления нейрона-победителя важной проблемой становится выбор метрики, в которой будет измеряться расстояние между векторами $x$ и $w_i$ . Чаще всего в качестве меры расстояния используются:

- **эвклидова мера** $$d(x,w_i)=\|x-w_i\|=\sqrt{\sum_{j=1}^{N}(x_j-w_{ij})^2} \quad ;(72)$$
- **скалярное произведение** $$d(x,w_i)=1-x\cdot w=1-\|x\|\cdot\|w_i\|\cdot\cos(x,w_i) \quad ;(73)$$
- **мера относительно нормы L1 (Манхэттен)** $$d(x,w_i)=\sqrt{\sum_{j=1}^{N}|x_j-w_{ij}|} \quad ;(74)$$
- **мера относительно нормы L∞** $$d(x,w_i)=\max_j|x_j-w_{ij}| \quad .(75)$$

При использовании эвклидовой меры разбиение пространства на зоны доминирования нейронов равносильно разбиению на области Вороного. При использовании другой меры формируется другое разделение областей влияния нейронов. Использование скалярного произведения требует нормализации входных векторов, так как в противном случае возможно неравномерное распределение нейронов: в одной области может находиться несколько нейронов, а в другой – ни одного нейрона.

# Нормализация векторов

При нормализованных обучающих векторах стремящиеся к ним векторы весов нормализуются автоматически. Однако нормализация векторов приводит к тому, что если $\|w_i\| = const$, то для всех нейронов при фиксированном значении $x$ произведение $\|x\|\|w_i\|$ также становится постоянной величиной. Следует отметить, что при нормализации вектора весов эвклидова мера и скалярное произведение равнозначны друг другу, так как $\|x - w_i\|^2 = \|x\|^2 + \|w_i\|^2 - 2x^T w_i$ .

Поэтому $\min\|x - w\|^2 = \max\left(x^T w_i\right)$ при $\|w_i\| = const$ .Экспериментальные исследования подтвердили необходимость применения нормализации векторов при малой размерности пространства. Такая нормализация проводится двумя способами:

- Переопределение компонентов вектора в соответствии с формулой

$$x_i \leftarrow \frac{x_i}{\sqrt{\sum_{i=1}^{N} x_i^2}} \quad . \qquad (76)$$

- Нормализация с использованием математического ожидания и дисперсии.

$$x_i \leftarrow \frac{x_i - Mx}{\sqrt{Dx}} \quad , \quad Mx = \frac{1}{n}\sum_{t=1}^{n} x_t \quad Dx = \frac{1}{n-1}\sum_{t=1}^{n}\left(x_t - Mx\right)^2 \qquad (77)$$

С увеличением размерности входного вектора эффект нормализации становится менее заметным, а при размерности $N > 200$ вообще сходит на нет.

# Проблема мертвых нейронов

При инициализации весов сети случайным способом часть нейронов может оказаться в области пространства, в которой отсутствуют данные или их количество ничтожно мало. Эти нейроны имеют мало шансов на победу и адаптацию своих весов, поэтому они остаются мертвыми. Поэтому важной проблемой становится активация всех нейронов сети, которую можно осуществить, если в алгоритме обучения предусмотреть учет количества побед каждого нейрона, а процесс обучения организовать так, чтобы дать шанс победить и менее активным нейронам.

Часто используется **метод подсчета потенциала $p_i$** каждого нейрона, значение которого модифицируется всякий раз после предъявления очередной реализации входного вектора $x$ в соответствии со следующей формулой (в ней предполагается, что победителем стал $w$-й нейрон):

$$p_i(t+1) = \begin{cases} p_i(t) + \frac{1}{K}; (i \neq w) \\ p_i(t) - p_{\min}; (i = w) \end{cases} . \quad (78)$$

Значение коэффициента $p_{\min}$ определяет минимальный потенциал, разрешающий участие в конкурентной борьбе. Если фактическое значение потенциала $p_i$ падает ниже , $p_{\min}$ , то $i$-й нейрон «отдыхает», а победитель ищется среди нейронов, для которых выполняется соотношение:

$$d(x, w_w) = \min\{d(x, w_i)\}, для 1 \le i \le K \; и \; p_i \ge p_{\min} \quad (79)$$

# Проблема мертвых нейронов

Максимальное значение потенциала ограничивается на уровне, равном 1. Выбор конкретного значения позволяет установить порог готовности нейрона к конкурентной борьбе. При $p_{\min}$ =0 утомляемость нейронов не возникает, и каждый из них сразу после победы будет готов к продолжению соперничества. При $p_{\min}$ =1 возникает другая крайность, вследствие которой нейроны побеждают по очереди, так как в каждый момент только один из них оказывается готовым к соперничеству. На практике хорошие результаты достигаются при $p_{\min}$ ≈0,74.

В другом очень удачном алгоритме обучения количество побед нейрона учитывается при **подсчете эффективного расстояния между вектором весов и реализацией обучающего вектора *x***. Это расстояние модифицируется пропорционально количеству побед данного нейрона в прошлом. Если обозначить количество побед *i*-го нейрона , такую модификацию можно представить в виде

$$d(x, w_i) \leftarrow N_i d(x, w_i) \qquad .(80)$$

Активные нейроны с большим значением штрафуются искусственным завышением этого расстояния. Отметим, что модификация расстояния производится только при выявлении победителя. В момент уточнения весов учитывается фактическое расстояние. Обычно после двух или трех циклов обучения модификация прекращается, что позволяет продолжить «честную» конкуренцию нейронов.

# Алгоритм WTA

Алгоритмы обучения, используемые для обучения нейронных сетей Кохонена, называются алгоритмами обучения без учителя. Подобные алгоритмы применяются в тех случаях, когда нет эталонных выходных значений для входных векторов.

Целью обучения сети Кохонена, считается такое упорядочение нейронов, которое минимизирует значение отклонения вектора весов от входного вектора *x*. При *p* входных векторах *x* эта погрешность в эвклидовой метрике может быть выражена в виде:

$$E = \frac{1}{2}\sum_{t=1}^{p}\left\|x_i - w_{w(t)}\right\| \quad ,(81)$$

где $w_{w(t)}$ - это вес нейрона-победителя при предъявлении вектора .

Этот подход также называется векторным квантованием (VQ). Номера нейронов-победителей при последовательном предъявлении векторов *x* образуют так называемую кодовую таблицу. Алгоритм ***WTA*(*WinnerTakesAll*** – победитель получает все) основан на данном подходе. В соответствии с ним после предъявления вектора *x* рассчитывается активность каждого нейрона. Победителем признается нейрон с самым сильным выходным сигналом, то есть тот, для которого скалярное произведение $(x^T w)$ оказывается наибольшим, что для нормализованных векторов равнозначно наименьшему эвклидову расстоянию между входным вектором и вектором весов нейронов. Победитель получает право уточнить свои веса в направлении вектора x согласно правилу

$$w_w(t+1) = w_w(t) + \eta(x - w_w(t)) \quad (82)$$

Веса остальных нейронов уточнению не подлежат. Алгоритм позволяет учитывать усталость нейронов путем подсчета количества побед каждого из них и поощрять элементы с наименьшей активностью для выравнивания их шансов.

# Алгоритм WTM

Помимо алгоритмов ***WTA***, в которых в каждой итерации может обучаться только один нейрон, для обучения сетей с самоорганизацией широко применяется алгоритмы типа ***WTM*** **(*WinnerTakesMost*** – победитель получает больше), в которых, кроме победителя, уточняют значения своих весов и нейроны из его ближайшего окружения. При этом, чем дальше какой-либо нейрон находится от победителя, тем меньше изменяются его веса. Процесс уточнения вектора весов может быть определен в виде

$$w_i(t+1) = w_i(t) + \eta_i S(i,x)(x - w_i(t)) \qquad (83)$$

для всех $i$ нейронов, расположенных в окрестности победителя. В приведенной формуле коэффициент обучения $\eta_i$ каждого нейрона отделен от его расстояния до вектора $x$ функцией $S(i,x)$ . Если $S(i,x)$ определяется в форме

$$S(i,x) = \begin{cases} 1, \text{для } i = w \\ 0, \text{для } i \neq w \end{cases}, \qquad (84)$$

где $w$ обозначает номер победителя, то мы получаем классический алгоритм *WTA*. Существует множество вариантов алгоритма *WTM*, отличающихся ,прежде всего формой функции $S(i,x)$ . Для дальнейшего обсуждения выберем классический алгоритм Кохонена.

# Алгоритм Кохонена

**Алгоритм Кохонена** относится к наиболее старым алгоритмам обучения сетей с самоорганизацией на основе конкуренции, и в настоящее время существуют различные его версии.

В классическом алгоритме Кохонена сеть инициализируется путем приписывания нейронам определенных позиций в пространстве и связывании их с соседями на постоянной основе. В момент выбора победителя уточняются не только его веса, но также веса и его соседей, находящихся в ближайшей окрестности. Таким образом, нейрон-победитель подвергается адаптации вместе со своими соседями.

$$S(i,x)=\begin{cases}1, для\ -K<d(i,w)<K\\ 0, иначе\end{cases},$$

(85)

В этом выражении $d(i,w)$ может обозначать как эвклидово расстояние между векторами весов нейрона-победителя $w$ и $i$-го нейрона, так и расстояние, измеряемое количеством нейронов.

# Алгоритм Кохонена

Другой тип соседства в картах Кохонена - это **соседство гауссового типа**, при котором функция соседства определяется формулой:

$$S(i,x) = \exp\left(-\frac{d^2(i,w)}{2\lambda^2}\right) . \quad (86)$$

Уточнение весов нейронов происходит по правилу:

$$w_i(t) = w_i(t-1) + \eta(t)\cdot S(i,w)\cdot(x - w_i) . \quad (87)$$

Степень адаптации нейронов-соседей определяется не только эвклидовым расстоянием между $i$-м нейроном и нейроном-победителем ($w$-м нейроном) $d(i,w)$, но также уровнем соседства $\lambda$.

Как правило, гауссово соседство, дает лучшие результаты обучения и обеспечивает лучшую организацию сети, чем прямоугольное соседство.

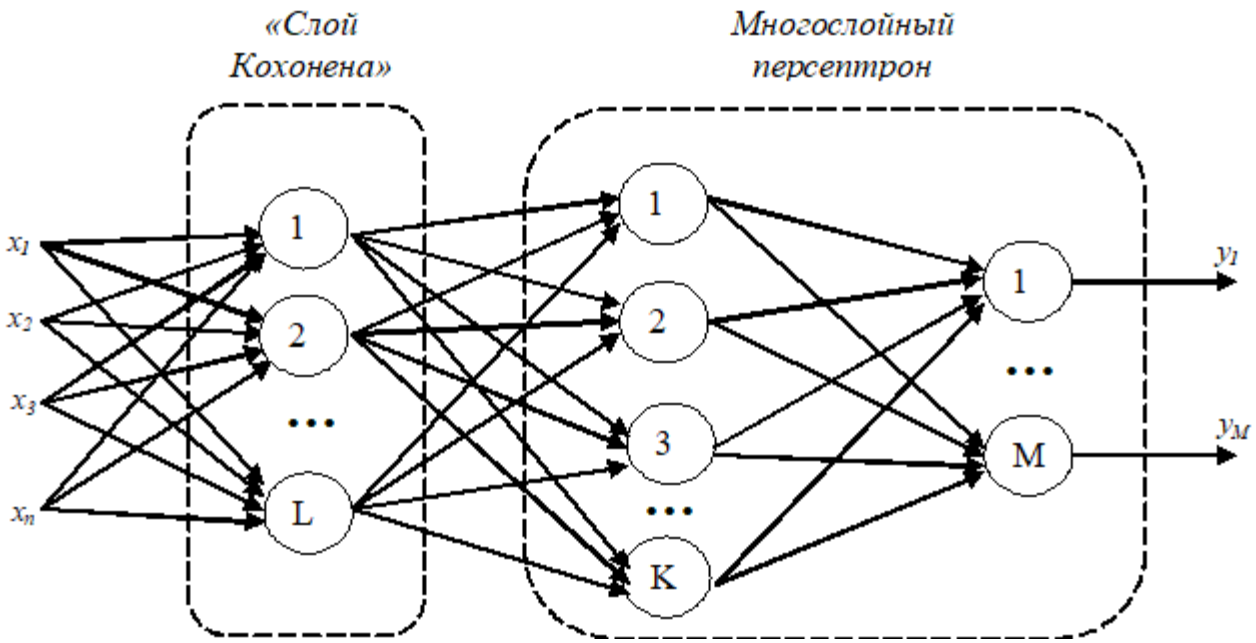
# Гибридная сеть Кохонена

Главной особенностью **сетей Кохонена** является очень высокая скорость обучения, много больше, чем у сетей, обучающихся с учителем. Так как сеть Кохонена выполняет обработку только входного вектора, то основным недостатком такой сети является отсутствие аппроксимирующих свойств, которыми обладают многослойный персептрон и радиально-базисные сети.

Очень хороший результат достигается при объединении самоорганизующегося слоя Кохонена и многослойного персептрона. Подобная структура, называемая гибридной сетью Кохонена, приведена на рисунке 13.

Слой Кохонена локализует значимые признаки процесса на основе анализа входного вектора $x$ , после чего формируется входной вектор персептронного слоя. Вследствие хорошей локализации признаков первым слоем, обычно достаточно использовать персептронную сеть с одним скрытым слоем нейронов.

# Гибридная сеть Кохонена (рисунок 13)



# Обучение гибридной сети

Обучение **гибридной сети** состоит из двух последовательных этапов. Сначала обучается слой Кохонена. В результате обучения векторы весов нейронов данного слоя с минимальной погрешностью отображают распределение данных обучающих векторов $x$. Обучение слоя Кохонена проводится по одному из алгоритмов для сети Кохонена. После завершения обучения, значения весов нейронов слоя Кохонена замораживаются, и производится анализ выходных сигналов. Выход нейрона-победителя переводится в состояние 1, а выходы остальных нейронов переводятся в состояния из интервала (0-1). Переход от фактических выходных сигналов к нормализованным сигналам может производиться по различным формулам. Хорошие результаты можно получить, используя формулу:

$$y_r = \exp\left(-\frac{\left|u_r - u_{\max}\right|^2}{\sigma^2}\right), \qquad (88)$$

где $u_r$ - реальный выход r-го нейрона слоя Кохонена, $y_r$ - нормализованный выход r-го нейрона слоя Кохонена, $u_{\max}$ - выход нейрона-победителя, $\sigma$ - параметр, подбираемый для конкретной задачи.

# Обучение гибридной сети

Затем обучается персептронная сеть по одному из алгоритмов обучения с учителем. Обучающими сигналами является множество пар $(y_t, d_t)$, где $y_t$ - нормализованный выходной вектор слоя Кохонена, а $d_t$ - это вектор эталонных значений выходных нейронов персептрона, соответствующих входному вектору $x_t$, $t = 1,2,..., p$.

Чаще всего используются градиентные алгоритмы и метод обратного распространения ошибки. Процесс обучения будет протекать гораздо быстрее, чем для обычного многослойного персептрона, благодаря хорошей локализации данных в слое Кохонена. Кроме того, при этом, как правило, достигается глобальный минимум функции погрешности.

Следует отметить, что данная сеть считается обобщением сети встречного распространения, в отличии от которой в сети Кохонена допускается дробная активность нейронов от значения 1 для победителя и от 0 до 1 для остальных нейронов. Учёт активности многих нейронов позволяет лучше локализовать входной вектор *x* в многомерном пространстве и получить лучшее отображение данных нейронной сетью в целом.

# Рекуррентные сети

Отдельную группу нейронных сетей составляют сети с обратной связью между различными слоями нейронов. Это так называемые **рекуррентные сети**. Их общая черта состоит в передаче сигналов с выходного либо скрытого слоя во входной слой. Главная особенность таких сетей – динамическая зависимость на каждом этапе функционирования. Изменение состояния одного нейрона отражается на всей сети вследствие обратной связи типа «один ко многим». В сети возникает **переходный процесс**, который завершается формированием нового устойчивого состояния, отличающегося в общем случае от предыдущего. Рекуррентные сети разделяются на 2 типа: рекуррентные сети на основе многослойного персептрона и самоорганизующиеся рекуррентные сети.

Особенностью самоорганизующихся рекуррентных сетей является тот факт, что для них не подходит ни **обучение с учителем**, ни обучение **без учителя**. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету.

С одной стороны, предъявление априорной информации можно расценивать, как помощь учителя, но с другой – сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с учителем не приходится.

# Рекуррентные сети

Из сетей с подобной логикой работы наиболее известны **сеть Хопфилда** и **сеть Хемминга**, которые использовались для организации ассоциативной памяти.

Главная задача ассоциативной памяти сводится к запоминанию входных обучающих выборок таким образом, чтобы при представлении новой выборки система могла сгенерировать ответ, – какая из запомненных ранее выборок наиболее близка к вновь поступившему образу. Наиболее часто в качестве меры близости отдельных векторов применяется мера Хемминга.

При использовании двоичных значений (0,1) расстояние Хемминга между двумя векторами $y = [y_1, y_2, ..., y_N]^T$ и $d = [d_1, d_2, ..., d_N]^T$ определяется в виде:

$$d_H(y,d) = \sum_{i=1}^{N} [d_i(1-y_i) + (1-d_i)y_i] \qquad (89)$$

При биполярных $(\pm 1)$ значениях элементов обоих векторов расстояние Хемминга рассчитывается по формуле:

$$d_H(y,d) = \frac{1}{2}\left[N - \sum_{i=1}^{N} y_i d_i\right] \qquad (90)$$

Мера Хемминга равна нулю только тогда, когда $y = d$. В противном случае она равна количеству битов, на которое различаются оба вектора.

# Сеть Хопфилда

Обобщенная структура сети Хопфилда представляется, как правило, в виде системы с непосредственной обратной связью выхода с входом (рисунок 14). Характерная особенность такой системы состоит в том, что выходные сигналы нейронов являются одновременно входными сигналами сети:

$$x_i(t) = y_i(t-1) \qquad (91)$$

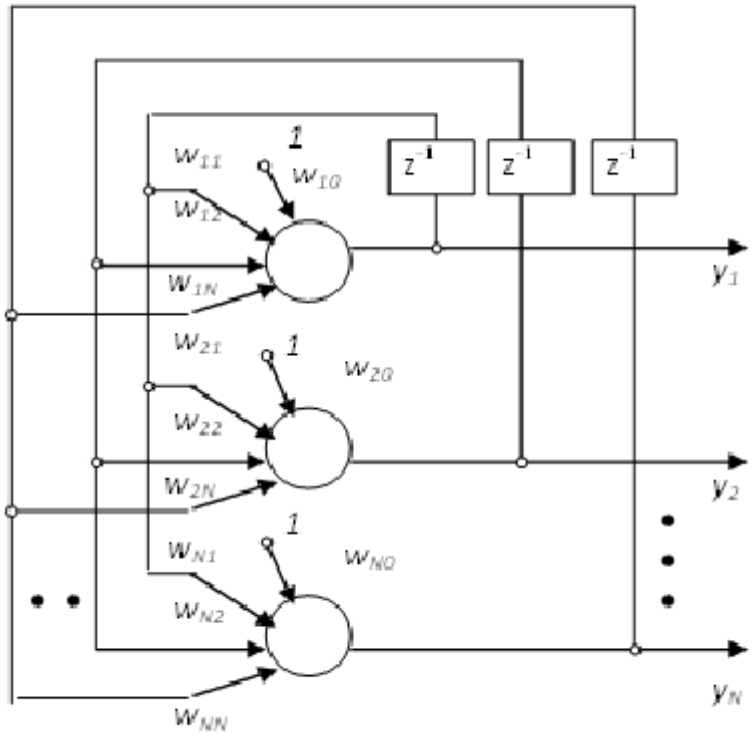
**В классической системе Хопфилда отсутствует связь нейрона с собственным выходом, что означает** $w_{ii} = 0$ , а матрица весов является симметричной $W = W^T$ . Процесс обучения сети формирует зоны притяжения некоторых точек равновесия, соответствующих обучающим данным.

Сеть Хопфилда состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, через который осуществляется ввод сигнала. В качестве функции активации нейронов сети Хопфилда будем использовать знаковую функцию, хотя для сетей Хопфилда также можно использовать пороговую функцию, линейную функцию с насыщением или сигмоидальные функции активации. Это означает, что выходной сигнал *i*-го нейрона определяется функцией:

$$y_i = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} x_j + b_i\right) \qquad ,(92)$$

где *N* обозначает число нейронов.

# Структура сети Хопфилда (рисунок 14)



# Сеть Хопфилда

Будем считать, что пороговые элементы являются компонентами вектора *x*. Без учета единичных задержек сети, представляющих собой способ синхронизации процесса передачи сигналов, основные зависимости, определяющие сеть Хопфилда, можно представить в виде:

$$y_i(t) = \operatorname{sgn}\left( \sum_{j=0, i \neq j}^{N} w_{ij} y_j(t-1) \right) \qquad (93)$$

с начальным условием $y_i(0) = x_j$ . В процессе функционирования сети Хопфилда можно выделить два режима: обучения и классификации. В режиме обучения на основе известных обучающих выборок *x* подбираются весовые коэффициенты $w_{ij}$ . В режиме классификации при зафиксированных значениях весов и вводе начального состояния $y(0) = x$ нейронов возникает переходной процесс, протекающий в соответствии с выражением (93) и завершающийся в одном из локальных минимумов, для которого $y(t) = y(t-1)$ .

При вводе только одной обучающей выборки $x_j$ процесс изменений продолжается до тех пор, пока зависимость (93) не начнет соблюдаться для всех нейронов. Это условие автоматически выполняется в случае выбора значений весов, соответствующих отношению

$$w_{ij} = \frac{1}{N} x_i x_j \qquad ,(94)$$

Следует отметить, что зависимость (94) представляет собой правило обучения Хебба.

# Сеть Хопфилда

При вводе большого числа обучающих выборок $x(t)$ для $t = 1, 2, ..., p$ веса подбираются согласно обобщенному правилу Хебба, в соответствии с которым

$$w_{ij} = \frac{1}{N} \sum_{t=1}^{p} x_i^{(t)} x_j^{(t)} \quad .(95)$$

Благодаря такому режиму обучения веса принимают значения, определяемые усреднением множества обучающих выборок.

Правило Хебба обладает невысокой продуктивностью. Максимальная емкость ассоциативной памяти (число запомненных образцов) при обучении по правилу Хебба с допустимой погрешностью 1%, составляет примерно 14% от числа нейронов сети. Кроме того, при наличии шума, применение правила Хебба приводит к различным неточностям в виде локальных минимумов, далеких от исходного решения. Поэтому в качестве альтернативы используют методы обучения, основанные на **псевдоинверсии**.

Идея этого метода состоит в том, что при правильно подобранных весах, каждая поданная на вход выборка *x* генерирует на выходе саму себя, мгновенно приводя к исходному состоянию (зависимость (93)) .

# Методы псевдоинверсии

В матричной форме это можно представить в виде:

$$WX = X \qquad (96)$$

где $W$ - матрица весов сети размерностью $N \times N$, а $X$ - прямоугольная матрица размерности $N \times p$, составленная из $p$ последовательных обучающих векторов $x^{(t)}$, то есть $X = \left[x^{(1)}, x^{(2)}, ..., x^{(p)}\right]$

Решение такой линейной системы уравнений имеет вид:

$$W = X\ X^{+} \quad , \qquad (97)$$

где знак + обозначает псевдоинверсию. Если обучающие векторы линейно независимы, последнее выражение можно представить в форме:

$$W = X\left(X^{T}\ X\right)^{-1} X^{T} \qquad (98)$$

Псевдоинверсия матрицы размерностью $N \times p$ в этом выражении заменена обычной инверсией квадратной матрицы $X^{T}\ X$ размерностью $p \times p$. Дополнительное достоинство выражения (98) – возможность записать его в итерационной форме, не требующей расчета обратной матрицы. В этом случае выражение (98) принимает вид функциональной зависимости от последовательности обучающих векторов $x^{(t)}$ для $t = 1, 2, ..., p$ :

# Методы псевдоинверсии

$$W^{(t)} = W^{(t-1)} + \frac{1}{\left[x^{(t)}\right]^T x^{(t)} - \left[x^{(t)}\right]^T W^{(t-1)} x^{(t)}} \times \left[W^{(t-1)} x^{(t)} - x^{(t)}\right] \times \left[W^{(t-1)} x^{(t)} - x^{(t)}\right] \quad (99)$$

при начальных условиях $W^{(0)} = 0$ . Такая форма предполагает однократное предъявление всех обучающих выборок, в результате чего матрица весов принимает значение $W = W^{(p)}$. Зависимости (98) и (99) называются методом

$\Delta-$ проекций. Применение метода псевдоинверсии увеличивает максимальную емкость сети Хопфилда, которая становится равной *N-1* .

Модифицированный вариант метода проекций – метод $\Delta-$ проекций – это градиентная форма алгоритма минимизации целевой функции. В соответствии с этим способом веса подбираются рекуррентно с помощью циклической процедуры, повторяемой для всех обучающих выборок:

$$W \leftarrow W + \frac{\eta}{N}\left[x^{(t)} - W x^{(t)}\right]\left[x^{(t)}\right]^T \quad . \quad (100)$$

Коэффициент $\eta$- это коэффициент обучения, выбираемый обычно из интервала [0.7 – 0.9]. Процесс обучения завершается, когда изменение вектора весов становится меньше априорно принятого значения $\varepsilon$.

# Методы псевдоинверсии

По завершении подбора весов сети их значения «замораживаются», и сеть можно использовать в режиме распознавания. В этой фазе на вход сети подается тестовый вектор  и рассчитывается ее отклик в виде:

$$y(t) = \mathrm{sgn}(W y(t-1)) \qquad (101)$$

(в начальный момент $y(0) = x$ ), причем итерационный процесс повторяется для последовательных значений $y(t)$ вплоть до стабилизации отклика.

В процессе распознавания образа по зашумленным сигналам, образующим начальное состояние нейронов, возникают проблемы с определением конечного состояния, соответствующего одному из запомненных образов. Возможны ошибочные решения. Одной из причин нахождения ошибочных решений является возможность перемешивания различных компонентов запомненных образов и формирования стабильного состояния, воспринимаемого как локальный минимум.

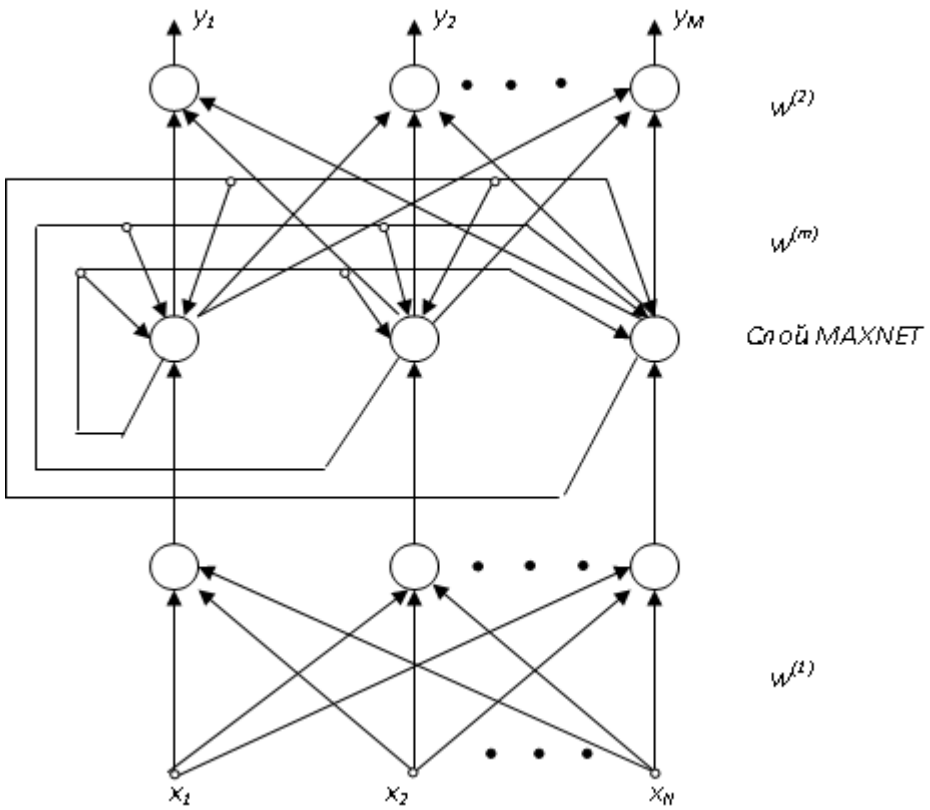
# Сеть Хемминга

**Сеть Хемминга** – это трехслойная рекуррентная структура, которую можно считать развитием сети Хопфилда, была предложена Р. Липпманом. Основная идея функционирования сети состоит в минимизации расстояния Хемминга между тестовым вектором, подаваемым на вход сети, и векторами обучающих выборок, закодированными в структуре сети. Обобщенная структура сети Хемминга представлена на рисунке 15.

Первый ее слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов. Второй слой, MAXNET, состоит из нейронов, связанных обратными связями по принципу «каждый с каждым», при этом в отличие от структуры Хопфилда существует ненулевая связь входа нейрона со своим собственным выходом. Веса нейронов в слое MAXNET постоянны. Разные нейроны связаны отрицательной обратной связью с весом $-\varepsilon$ , при этом обычно величина $\varepsilon$ обратно пропорциональна числу образов.

С собственным выходом нейрон связан положительной обратной связью с весом +1. Веса пороговых элементов равны нулю. Нейроны этого слоя функционируют в режиме WTA, при котором в каждой фиксированной ситуации активизируется только один нейрон.

# Структура сети Хемминга (рисунок 15)



# Функционирование сети Хемминга

Выходной однонаправленный слой формирует выходной вектор $y^{(t)}$, в котором только один нейрон имеет выходное значение, равное 1, а все остальные – равные 0.

Веса первого слоя соответствуют входным векторам-образцам $x^{(t)}$, поэтому для $t=1,2,...,p$, $j=1,2,...,N$, $i=t$, $w_{ij}^{(1)}=x_j^{(t)}$, то есть веса первого нейрона запоминают компоненты первого входного вектора. Веса второго нейрона – компоненты второго вектора и т. д.

Аналогично веса выходного слоя соответствуют очередным векторам образов $w_{li}^{(2)}=y_l^{(t)}$, $i,t,l=1,2,...p$.

В случае нейронов слоя **MAXNET**, функционирующих в режиме WTA, веса сети должны усиливать собственный сигнал нейрона и ослаблять остальные сигналы. Для достижения этого принимается $w_{ii}^{(m)}=1$, а также

$$-\frac{1}{p-1}<w_{is}^{(m)}<0 \qquad (102)$$

для $i\neq s$. Для обеспечения абсолютной сходимости алгоритма веса должны отличать друг от друга. Р. Липпман в своей работе принял

$$w_{is}^{(m)}=-\frac{1}{p-1}+\xi \quad , \qquad (103)$$

# Функционирование сети Хемминга

где $\xi$ - случайная величина с достаточно малой амплитудой.

В процессе функционирования сети в режиме распознавания можно выделить три фазы. В первой из них на вход подается *N*-элементный вектор *x*. После предъявления этого вектора на выходах нейронов первого слоя генерируются сигналы, задающие начальные состояния нейронов второго слоя. Нейроны первого слоя рассчитывают расстояния Хемминга между поданными на вход сети вектором *x* и векторами весов отдельных нейронов этого слоя. Значения выходных сигналов этих нейронов определяются по формуле:

$$\hat{y}_i = 1 - \frac{d_H\left(x^{(t)}, x\right)}{N}, \qquad (104)$$

где $d_H\left(x^{(t)}, x\right)$ обозначает расстояние Хемминга между входными векторами $x^{(t)}$ и $x$, то есть число битов, на которое различаются эти два вектора. Значение $\hat{y}_i = 0$, если $x = -x^{(t)}$, и $\hat{y}_i = 1$, если $x = x^{(t)}$.

В остальных случаях значения лежат в интервале [0, 1].

Сигналы $\hat{y}_i$ нейронов первого слоя становятся начальными состояниями нейронов слоя MAXNET на второй фазе функционирования сети.

# Функционирование сети Хемминга

Во второй фазе инициировавшие MAXNET сигналы удаляются, и из сформированного ими начального состояния запускается итерационный процесс. Итерационный процесс завершается в момент, когда все нейроны, кроме нейрона-победителя с выходным сигналом не равным 0, перейдут в нулевое состояние.

Задача нейронов этого слоя состоит в определении победителя, то есть нейрона, у которого выходной сигнал отличен от 0. Процесс определения победителя выполняется согласно формуле:

$$y_l(t) = f\left(\sum_s w_{is}^{(m)} y_s(t-1)\right) = f\left(y_l(t-1) + \sum_{s \neq i} w_{is}^{(m)} y_s(t-1)\right) \quad ,(105)$$

при начальном значении $y_s(0) = \hat{y}_i$. Функция активации $f(y)$ нейронов слоя MAXNET задается выражением:

$$f(y) = \begin{cases} y \text{ для } y \geq 0 \\ 0 \text{ для } y < 0 \end{cases} \quad (106)$$

Итерационный процесс (105), завершается в момент, когда состояние нейронов стабилизируется, и активность продолжает проявлять только один нейрон, тогда как остальные пребывают в нулевом состоянии. Активный нейрон становится победителем и через веса $w_{is}^{(2)}$ $(s = 1,2,..,N)$ линейных нейронов выходного слоя представляет вектор $y^{(t)}$, который соответствует номеру вектора $x^{(t)}$, признанному слоем MAXNET в качестве ближайшего к входному вектору $x$.

# Функционирование сети Хемминга

В третьей фазе этот нейрон посредством весов, связывающих его с нейронами выходного слоя, формирует на выходе сигнал, равный 1, его номер является номер входного образца, к которому принадлежит входной вектор.

Входные узлы сети принимают значения, задаваемые аналогичными компонентами вектора $x$. Нейроны первого слоя рассчитывают расстояние Хемминга между входным вектором $x$ и каждым из $p$ закодированных векторов-образцов $x^{(t)}$, образующих веса нейронов первого слоя. Нейроны в слое MAXNET выбирают вектор с наименьшим расстоянием Хемминга, определяя, таким образом, класс, к которому принадлежит предъявленный входной вектор x. Веса нейронов выходного слоя формируют вектор, соответствующий классу входного вектора. При $p$ нейронах первого слоя, емкость запоминающего устройства Хемминга также равна $p$, так как каждый нейрон представляет единственный класс.

Важным достоинством сети Хемминга считается небольшое, по сравнению с сетью Хопфилда, число взвешенных связей между нейронами. Так, например, 100-входовая сеть Хопфилда, кодирующая 10 различных векторных классов, должна содержать 10000 взвешенных связей, тогда как аналогичная сеть Хемминга содержит 1100 связей, из которых 1000 весов находятся в первом слое, а 100 – в слое MAXNET.

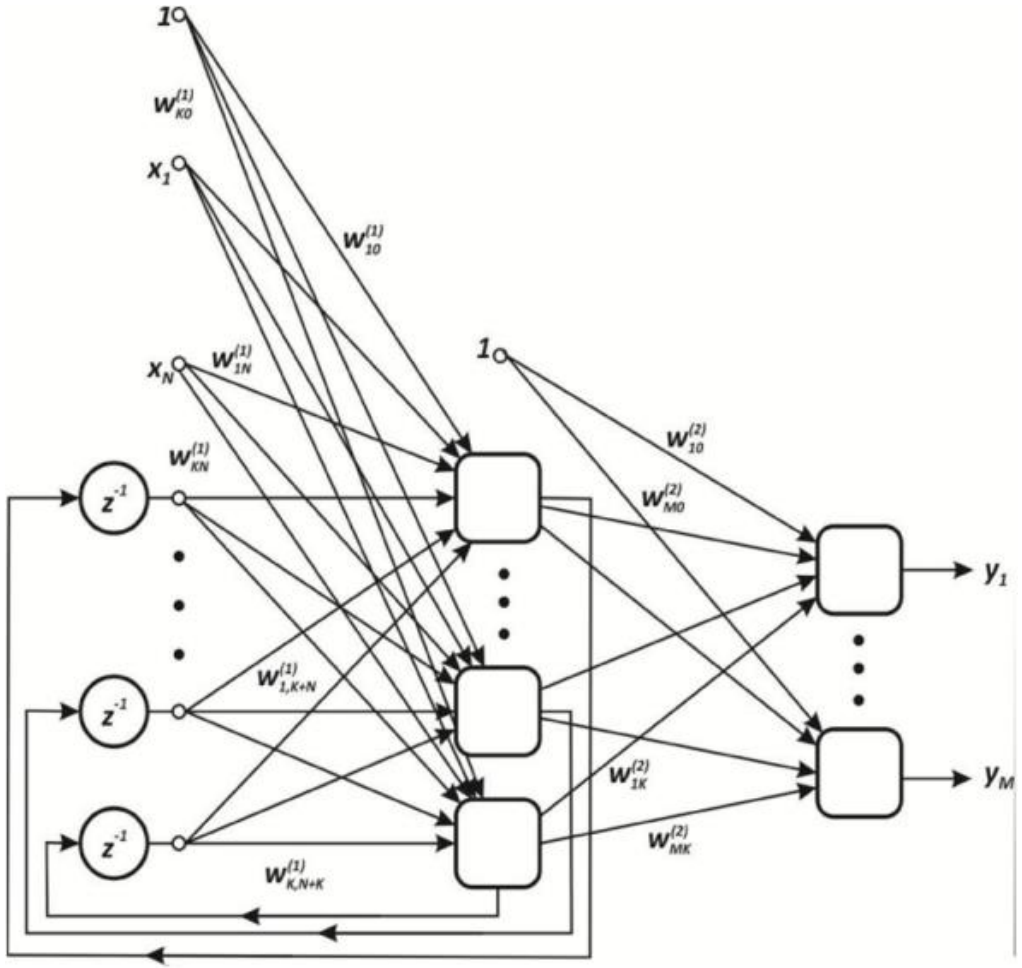
# Рекуррентная сеть Эльмана

Данная рекуррентная сеть представляет собой развитие сетей персептронного типа за счет добавления в них обратных связей. **Сеть Эльмана** является одним из представителей типа сетей, названных **рекуррентными многослойными персептронами (RMLP).** Рекуррентные персептроны хорошо справляются не только с задачей классификации, но и с задачей прогнозирования.

Сеть Эльмана характеризуется **частичной рекуррентностью** в форме обратной связи между входным и скрытым слоем, реализуемой с помощью единичных элементов запаздывания $z^{-1}$ . Обобщенная структура этой сети представлена на рисунке 16. Каждый скрытый нейрон имеет свой аналог в контекстном слое, образующем совместно с внешними входами сети входной слой. Выходной слой состоит из нейронов, однонаправленно связанных только с нейронами скрытого слоя. Обозначим внутренний вектор возбуждения сети $x$ (в его состав входит пороговый элемент), состояния скрытых нейронов - $\nu \in R^K$ , а выходные сигналы сети - $y \in R^M$ . Тогда входной вектор сети в момент времени $t$ имеет форму:

$$x(t) = \left[x_0(t), x_1(t), \ldots, x_N(t), \nu_1(t-1), \nu_2(t-1), \ldots, \nu_K(t-1)\right] \qquad (107)$$

# Структура рекуррентной сети Эльмана (рисунок 16)



# Рекуррентная сеть Эльмана

Веса синаптических связей первого (скрытого) слоя сети обозначим $w_{ij}^{(1)}$, а второго (выходного) слоя $w_{si}^{(2)}$. Если взвешенную сумму i-го нейрона скрытого слоя обозначить $u_i$, а его выходной сигнал - $v_i$, то

$$u_i(t) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(t) \qquad (108)$$

$$v_i(t) = f_1(u_i(t)) \qquad ;(109)$$

Веса $w_{ij}^{(1)}$ образуют матрицу $W^{(1)}$ синаптических связей скрытого слоя, а $f_1(u_i)$ - функция активации i-го нейрона скрытого слоя. Аналогично можно обозначить взвешенную сумму s-го нейрона выходного слоя $g_s$, а соответствующий ему выходной сигнал сети - $y_s$. Эти сигналы описываются формулами:

$$g_s(t) = \sum_{i=0}^{K} w_{si}^{(2)} v_s(t) \qquad ;(110)$$

.

$$y_s(t) = f_2(g_s(t)) \qquad (111)$$

В свою очередь, веса $w_{si}^{(2)}$ образуют матрицу $W^{(2)}$, описывающую синаптические связи нейронов выходного слоя, а $f_2(u_s)$ - функция активации s-го нейрона выходного слоя.

# Алгоритм обучения сети Эльмана

Для обучения сети Эльмана будем использовать градиентный метод наискорейшего спуска.

Для этого метода необходимо задать формулы, позволяющие рассчитывать градиент целевой функции в текущий момент времени. Целевая функция в момент времени $t$ определяется как сумма квадратов разностей между значениями выходных сигналов сети и их ожидаемыми значениями для всех $M$ выходных нейронов:

$$E(t)=\frac{1}{2}\sum_{s=1}^{M}\left[y_s(t)-d_s(t)\right]^2=\frac{1}{2}\sum_{s=1}^{M}e_s(t)^2 \qquad (112)$$

При дифференцировании целевой функции относительно весов выходного слоя получаем:

$$\nabla^{(2)}_{\alpha\beta}E(t)=\frac{\partial E(t)}{\partial w^{(2)}_{\alpha\beta}}=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\frac{dg_s(t)}{dw^{(2)}_{\alpha\beta}}=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=0}^{K}\frac{d\left(w^{(2)}_{si}v_i(t)\right)}{dw^{(2)}_{\alpha\beta}}= \qquad (113)$$

$$\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=0}^{K}\left(\frac{dv_i}{dw^{(2)}_{\alpha\beta}}w^{(2)}_{si}+\frac{dw^{(2)}_{si}}{dw^{(2)}_{\alpha\beta}}v_i(t)\right)$$

Связи между скрытым и выходным слоем однонаправленные, поэтому:

$$\frac{dv_i}{dw^{(2)}_{\alpha\beta}}=0 \qquad (114)$$

# Алгоритм обучения сети Эльмана

С учетом этого факта получим:

$$\nabla^{(2)}_{\alpha\beta}E(t)=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=0}^{K}\frac{dw^{(2)}_{si}}{dw^{(2)}_{\alpha\beta}}v_i(t)=e_\alpha(t)\frac{df_2(g_\alpha(t))}{dg_\alpha(t)}v_\beta(t) \quad (115)$$

При использовании метода наискорейшего спуска веса уточняются по формулам:

$$w^{(2)}_{\alpha\beta}(t+1)=w^{(2)}_{\alpha\beta}(t)+\Delta w^{(2)}_{\alpha\beta} \quad (116)$$

$$\Delta w^{(2)}_{\alpha\beta}=-\eta\nabla^{(2)}_{\alpha\beta}E(t) \quad (117)$$

Формулы уточнения весов скрытого слоя сети Эльмана более сложные по сравнению с персептронной сетью из-за наличия обратных связей между скрытым и контекстным слоями. Расчет компонентов вектора градиента целевой функции относительно весов скрытого слоя реализуется по формулам:

$$\nabla^{(1)}_{\alpha\beta}E(t)=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=1}^{K}\frac{d(v_i w^{(2)}_{si})}{dw^{(1)}_{\alpha\beta}}=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=1}^{K}\frac{dv_i(t)}{dw^{(1)}_{\alpha\beta}}w^{(2)}_{si} \quad (118)$$

$$\frac{dv_i(t)}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_i)}{du_i}\sum_{k=0}^{N+K}\frac{d(x_k(t)w^{(1)}_{ik})}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_i)}{du_i}\left[\delta_{i\alpha}x_\beta+\sum_{k=N+1}^{N+K}\frac{dx_k(t)}{dw^{(1)}_{\alpha\beta}}w^{(1)}_{ik}\right] \quad (119)$$

где - дельта Кронекера, то есть: $\delta_{i\alpha}=\begin{cases}1, i=\alpha\\ 0, i\neq\alpha\end{cases}$

.

# Алгоритм обучения сети Эльмана

Из определения входного вектора (формула (107)) в момент времени $t$ следует выражение (120):

$$\frac{d\nu_i(t)}{dw_{\alpha\beta}^{(1)}} = \frac{df_1(u_i)}{du_i}\left[\delta_{i\alpha}x_\beta + \sum_{k=N+1}^{N+K}\frac{d\nu_k(t-1)}{dw_{\alpha\beta}^{(1)}}w_{ik}^{(1)}\right] = \frac{df_1(u_i)}{du_i}\left[\delta_{i\alpha}x_\beta + \sum_{k=1}^{K}\frac{d\nu_k(t-1)}{dw_{\alpha\beta}^{(1)}}w_{i,k+N}^{(1)}\right] \quad (120)$$

Это выражение позволяет рассчитать производные целевой функции относительно весов скрытого слоя в момент времени $t$ . Следует отметить, что это рекуррентная формула, определяющая производную в момент времени $t$ в зависимости от ее значения в предыдущий момент *t-1*. Начальные значения производных в момент $t = 0$ считаются нулевыми:

$$\frac{d\nu_1(0)}{dw_{\alpha\beta}^{(1)}} = \frac{d\nu_2(0)}{dw_{\alpha\beta}^{(1)}} = \ldots = \frac{d\nu_K(0)}{dw_{\alpha\beta}^{(1)}} = 0 \quad (121)$$

Таким образом, алгоритм обучения сети Эльмана можно представить в следующем виде:

1. Присвоить весам случайные начальные значения, имеющие, как правило, равномерное распределение в определенном интервале (например, между -1 и 1).
2. Для очередного момента $t$ $(t = 0,\ 1,\ 2,\ldots)$ определить состояние всех нейронов сети (сигналы $\nu_l$ и $y_l$ ). На этой основе можно сформировать входной вектор $x(t)$ для произвольного момента $t$ .

# Алгоритм обучения сети Эльмана

3. Определить вектор погрешности обучения $e(t)$ для нейронов выходного слоя как разность между фактическим и ожидаемым значениями сигналов выходных нейронов.
4. Сформировать вектор градиента целевой функции относительно весов выходного и скрытого слоя с использованием формул (115), (118) и (119).
5. Уточнить значения весов сети согласно правилам метода наискорейшего спуска:

- для нейронов выходного слоя сети по формуле

$$w_{\alpha\beta}^{(2)}(t) = w_{\alpha\beta}^{(2)}(t-1) - \eta\nabla_{\alpha\beta}^{(2)}E(t) \qquad (122)$$

- для нейронов скрытого слоя сети по формуле

$$w_{\alpha\beta}^{(1)}(t) = w_{\alpha\beta}^{(1)}(t-1) - \eta\nabla_{\alpha\beta}^{(1)}E(t) \qquad (123)$$

После уточнения значений весов перейти к пункту 2 алгоритма для расчета в очередной момент времени .

Практические реализации алгоритма обучения сети Эльмана строятся на методе наискорейшего спуска, усиленном моментом. Это значительно повышает эффективность обучения и вероятность достижения глобального минимума целевой функции.

# Алгоритм обучения сети Эльмана

При использовании такого подхода уточнение вектора весов в момент времени выполняется в соответствии с формулой:

$$\Delta w(t) = -\eta \nabla E(t) + \alpha(t)\Delta w(t-1) \quad ,(124)$$

где $\alpha(t)$ - коэффициент момента, выбираемый из интервала (0, 1). Первое слагаемое этого выражения соответствует обычному методу обучения, второе – учитывает фактор момента, отражающий последнее изменение весов и не зависящий от фактического значения градиента. Чем больше величина $\alpha(t)$ , тем большее влияние на подбор весов оказывает слагаемое момента. Его значение существенно возрастает на плоских участках целевой функции и около локального минимума, где значение градиента близко к нулю.

В окрестностях локального минимума фактор момента может вызвать изменение весов, ведущее к росту целевой функции и к выходу из зоны локального минимума с возобновлением поиска глобального минимума.

# Сеть Вольтерри

**Сеть Вольтерри** относится к сетям со специализированной структурой. Хорошо подходит для решения задач прогнозирования и аппроксимации. Это динамическая сеть для нелинейной обработки последовательности сигналов, задержанных относительно друг друга. Возбуждением для сети в момент $t$ служит вектор $x = [x_t, x_{t-1}, ..., x_{t-L}]^T$, где $L$ - количество единичных задержек, а $L+1$ означает длину вектора. В соответствии с определением ряда Вольтерри выходной сигнал генерируется по формуле:

$$y(t) = \sum_{i_1=1}^{L} w_{i_1} x(t-i_1) + \sum_{i_1=1}^{L}\sum_{i_2=1}^{L} w_{i_1 i_2} x(t-i_1)x(t-i_2) + ... + \sum_{i_1=1}^{L} ... \sum_{i_K=1}^{L} w_{i_1 i_2 ... i_K} x(t-i_1)x(t-i_2)...x(t-i_K) \quad (125)$$

где $x$ обозначает входной сигнал, а веса $w_{i_1}, w_{i_2}, ..., w_{i_1 i_2 ... i_K}$, называемые **ядрами Вольтерри**, соответствуют реакциям высших порядков. **Порядок полинома Вольтерри $K$ также называется степенью ряда Вольтерри.**

Нелинейная функциональная зависимость Вольтерри является полиномиальным обобщением описания линейного фильтра *FIR*. Порядок этого полинома $K$ называется степенью ряда Вольтерри. Пусть целевая функция для одного обучающего вектора выражается формулой:

$$E = \frac{1}{2}(y(t) - d(t))^2 \quad (126)$$

# Сеть Вольтерри

В этом случае можно минимизировать значение целевой функции градиентными методами, которые сводятся к расчёту приращения весов нейронов при помощи системы дифференциальных уравнений вида:

$$\frac{dw_{i_1}}{dt} = -\mu(y(t) - d(t))x(t - i_1),$$
$$\frac{dw_{i_1 i_2}}{dt} = -\mu(y(t) - d(t))x(t - i_1)x(t - i_2),$$
$$\frac{dw_{i_1 i_2 i_3}}{dt} = -\mu(y(t) - d(t))x(t - i_1)x(t - i_2)x(t - x_3),$$
$$\frac{dw_{i_1 i_2 \ldots i_{K-1} i_K}}{dt} = -\mu(y(t) - d(t))x(t - i_1)x(t - i_2)\ldots x(t - i_k) \qquad (127)$$

Для упрощения структуры сети представленное разложение Вольтерри можно записать в следующей форме:

$$y_t = \sum_{i_1=0}^{L} x_{t-1}\left[ w_{i_1} + \sum_{i_2=0}^{L} x_{t-i_2}\left[ w_{i_1 i_2} + \sum_{i_3=0}^{L} x_{t-i_3}(w_{i_1 i_2 i_3} + \ldots) \right] \right] \qquad (128)$$

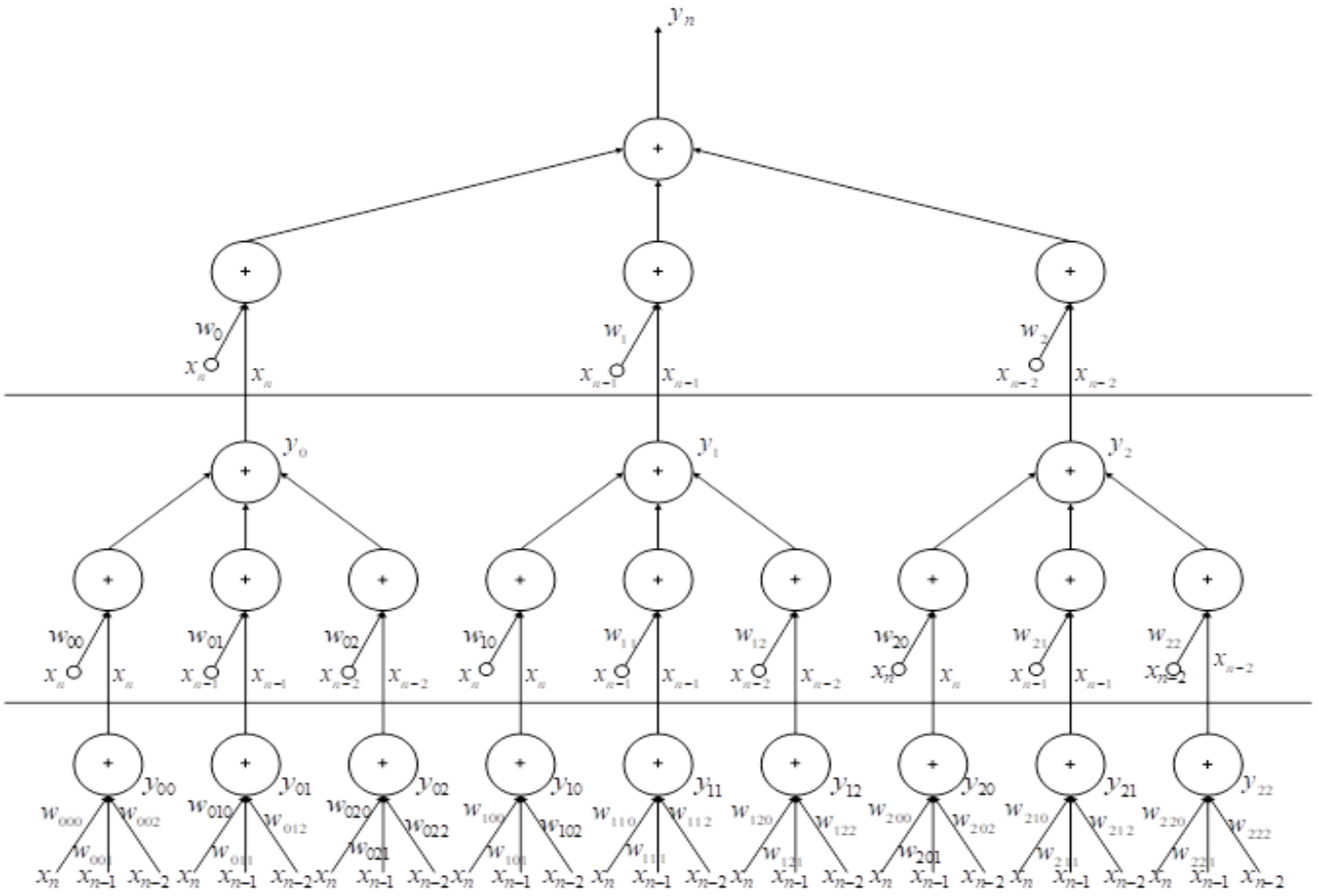
# Сеть Вольтерри

где используются обозначения $y_t = y(t), x_{t-i_1} = x(t-i_1)$ и т.д.

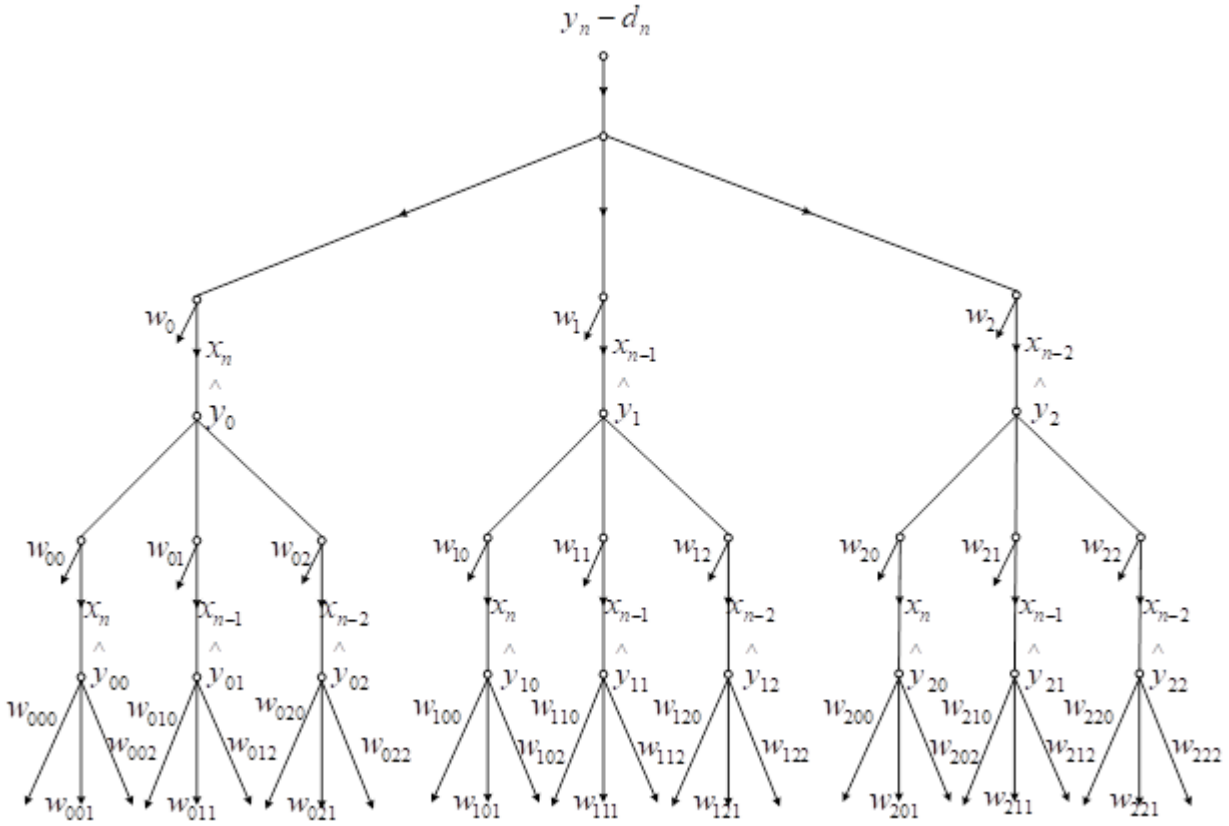
Каждое слагаемое в квадратных скобках представляет собой линейный фильтр первого порядка, в котором соответствующие веса представляют импульсную реакцию другого линейного фильтра следующего уровня. Количество слоёв сети Вольтерри равно порядку полинома $K$. На рисунке 16 показано распространение сигналов по сети Вольтерри, реализующей зависимость (128) при количестве единичных задержек $L$=2 , $K = 3$ .

Система представляет собой структуру типичной многослойной однонаправленной динамической нейронной сети. Это сеть с полиномиальной нелинейностью. Подбор весов производится последовательно слой за слоем, причём эти процессы независимы друг от друга, и, увеличение числа весов в слое и числа самих слоёв в незначительной степени сказывается на обусловленности задачи. Это даёт возможность существенно увеличить длину $L$ и порядок системы $K$ при её практической реализации. Обучение нейронной сети Вольтерри лучше всего приводить с использованием технологии сопряжённых графов. Для сети, представленной на рисунке 17, сопряжённый граф строится без особого труда (рисунок 18).

# Граф сети Вольтерри (рисунок 17)



# Сопряжённый граф сети Вольтерри (рисунок 18)



# Сеть Вольтерри

В соответствии с обозначениями, принятыми на рисунках, возбуждением сопряженного графа служит разностный сигнал $(y_t - d_t)$ , где $d_t$ обозначает ожидаемое, а $y_t$ - фактическое значение в выходном узле системы в момент *t*. Формулы для определения конкретных компонентов вектора градиента имеют вид:

$$\frac{\partial E}{\partial w_{i_1}} = x_{t-i_1}(y_t - d_t),$$

$$\frac{\partial E}{\partial w_{i_1 i_2}} = x_{t-i_2}\hat{y}_{i_1},$$

$$\frac{\partial E}{\partial w_{i_1 i_2 i_3}} = x_{t-i_3}\hat{y}_{i_1 i_2},$$

$$\frac{\partial E}{\partial w_{i_1 i_2 \ldots i_{K-1} i_K}} = x_{t-i_k}\hat{y}_{i_1 i_2 \ldots i_{K-1}}$$

$$\hat{y}_{i_1} = (x_{t-i_1}(y_t - d_t)) \qquad (129)$$

$$\hat{y}_{i_1 i_2} = (x_{t-i_2}\hat{y}_{i_1})$$

$$\hat{y}_{i_1 i_2 i_3} = (x_{t-i_3}\hat{y}_{i_2}))$$

# Сеть Вольтерри

В приведенных формулах сигналы, обозначенные символом ^, соответствуют сопряженному, а остальные – исходному графу системы. После определения конкретных компонентов градиента обучение сети с применением оптимизационного метода наискорейшего спуска может быть сведено к решению дифференциальных уравнений:

$$\frac{dw_{i_1}}{dt} = -\mu \frac{\partial E}{\partial w_{i1}},$$

$$\frac{dw_{i_1 i_2}}{dt} = -\mu \frac{\partial E}{\partial w_{i_1 i_2}},$$

$$\frac{dw_{i_1 i_2 i_3}}{dt} = -\mu \frac{\partial E}{\partial w_{i_1 i_2 i_3}},$$

$$\frac{dw_{i_1 i_2 \ldots i_{K-1} i_K}}{dt} = -\mu \frac{\partial E}{\partial w_{i_1 i_2 \ldots i_{K-1} i_K}}. \qquad (130)$$

где $\mu$ - коэффициент обучения.

# Сеть Вольтерри

Важным достоинством метода сопряженных графов считается простота учета равных значений весов в различных ветвях сети. Симметрия ядер Вольтерри приводит к равенству весов $w_{i_1 i_2 \ldots i_K}$ для всех перестановок индексов $i_1, i_2, \ldots, i_K$. Для двухиндексных весов это означает, что $w_{i_1 i_2} = w_{i_2 i_1}$. Подобные соотношения можно написать и для трёхиндексных и четырёхиндексных и так далее весов. Анализ обозначений весов сети, изображённой на рисунке 16, позволяет найти веса, которые имеют одни и те же значения.

С учетом симметрии ядер Вольтерри выражения, описывающие компоненты градиента относительно весов $w_{i_1 i_2 \ldots i_K}$, могут быть определенным образом модифицированы:

$$\frac{\partial E}{\partial w_{i_1 i_2}} = x_{t-i_2}\hat{y}_{i_1} + x_{t-i_1}\hat{y}_{i_2}, \qquad (131)$$

$$\frac{\partial E}{\partial w_{i_1 i_2 i_3}} = x_{t-i_3}\hat{y}_{i_1 i_2} + x_{t-i_2}\hat{y}_{i_1 i_3} + x_{t-i_3}\hat{y}_{i_2 i_1} + x_{t-i_1}\hat{y}_{i_2 i_3} + x_{t-i_2}\hat{y}_{i_3 i_1} + x_{t-i_1}\hat{y}_{i_3 i_2}$$