

Нейронные Сети

STATISTICA Neural Networks

Передовые технологии
анализа данных



data analysis • data mining • quality control • web-based analytics

U.S. Headquarters: StatSoft, Inc. • 2300 E. 14th St. • Tulsa, OK 74104 • USA • (918) 749-1119 • Fax: (918) 749-2217 • info@statsoft.com • www.statsoft.com

Australia: StatSoft Pacific Pty Ltd.

Brazil: StatSoft South America

Bulgaria: StatSoft Bulgaria Ltd.

Czech Rep.: StatSoft Czech Rep. s.r.o.

China: StatSoft China

France: StatSoft France

Germany: StatSoft GmbH

Hungary: StatSoft Hungary Ltd.

India: StatSoft India Pvt. Ltd.

Israel: StatSoft Israel Ltd.

Italy: StatSoft Italia srl

Japan: StatSoft Japan Inc.

Korea: StatSoft Korea

Netherlands: StatSoft Benelux BV

Norway: StatSoft Norway AS

Poland: StatSoft Polska Sp. z o.o.

Portugal: StatSoft Ibérica Lda

Russia: StatSoft Russia

Spain: StatSoft Ibérica Lda

S. Africa: StatSoft S. Africa (Pty) Ltd.

Sweden: StatSoft Scandinavia AB

Taiwan: StatSoft Taiwan

UK: StatSoft Ltd.

Теоретические основы нейронных сетей



- Биологическая модель нейрона
- Математические модели нейрона
- Нейронные сети как ориентированные графы
- Архитектура сетей
- Принципы обучения нейронных сетей
- Однослойный Персептрон
- Многослойный Персептрон
- Сети на основе Радиальных Базисных Функций (РБФ)
- Вероятностные и Обобщенно-Регрессионные сети
- Карты самоорганизации
- Ансамбли сетей

Что такое нейронные сети

- Мозг состоит из простейших клеток – нейронов
- Нейрон – элементарная структурная единица обработки информации
- Вопрос: как эти простейшие клетки могут перерабатывать сложную информацию?
- Очевидно, из простейших нейронов можно собрать довольно сложную конструкцию
- Вопрос: как мозг организует нейроны, чтобы они могли выполнять конкретные задачи (зрение, эхолокация) во много раз быстрее современных компьютеров?

Что такое нейронные сети

- Биологические модели мозга привели к математическим моделям
- Искусственная нейронная сеть – машина, моделирующая способ обработки мозгом конкретной задачи

Человеческий Мозг

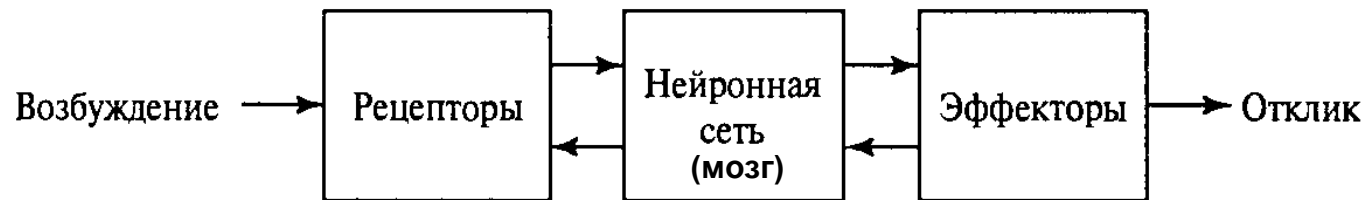


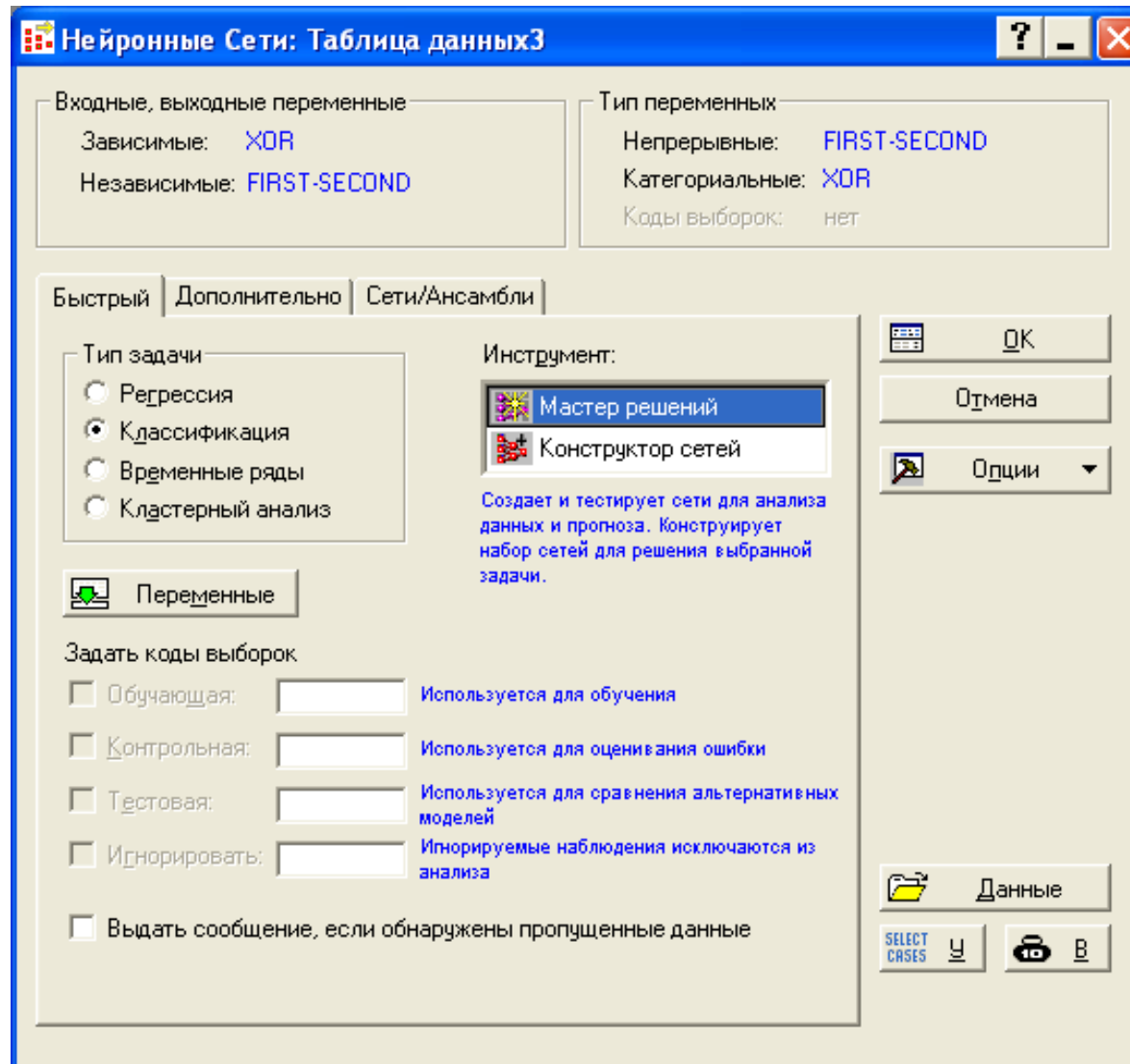
Рис. 1.1. Блочная диаграмма для нервной системы

- Нейроны – простейшие клетки мозга
- Синапсы – обеспечивают связи между нейронами
- Пластичность – способность нервной системы адаптироваться к условиям окружающей среды

Примеры обработки информации

- Классификация – ребенку предлагают картинки, и он учится различать круги и квадраты
- Прогнозирование – человек попадает в схожие ситуации, учится прогнозировать их развитие, т.е. предсказать будущее
- Сложные зависимости или связи – человеческая речь

Методы решения таких задач реализованы в STATISTICA



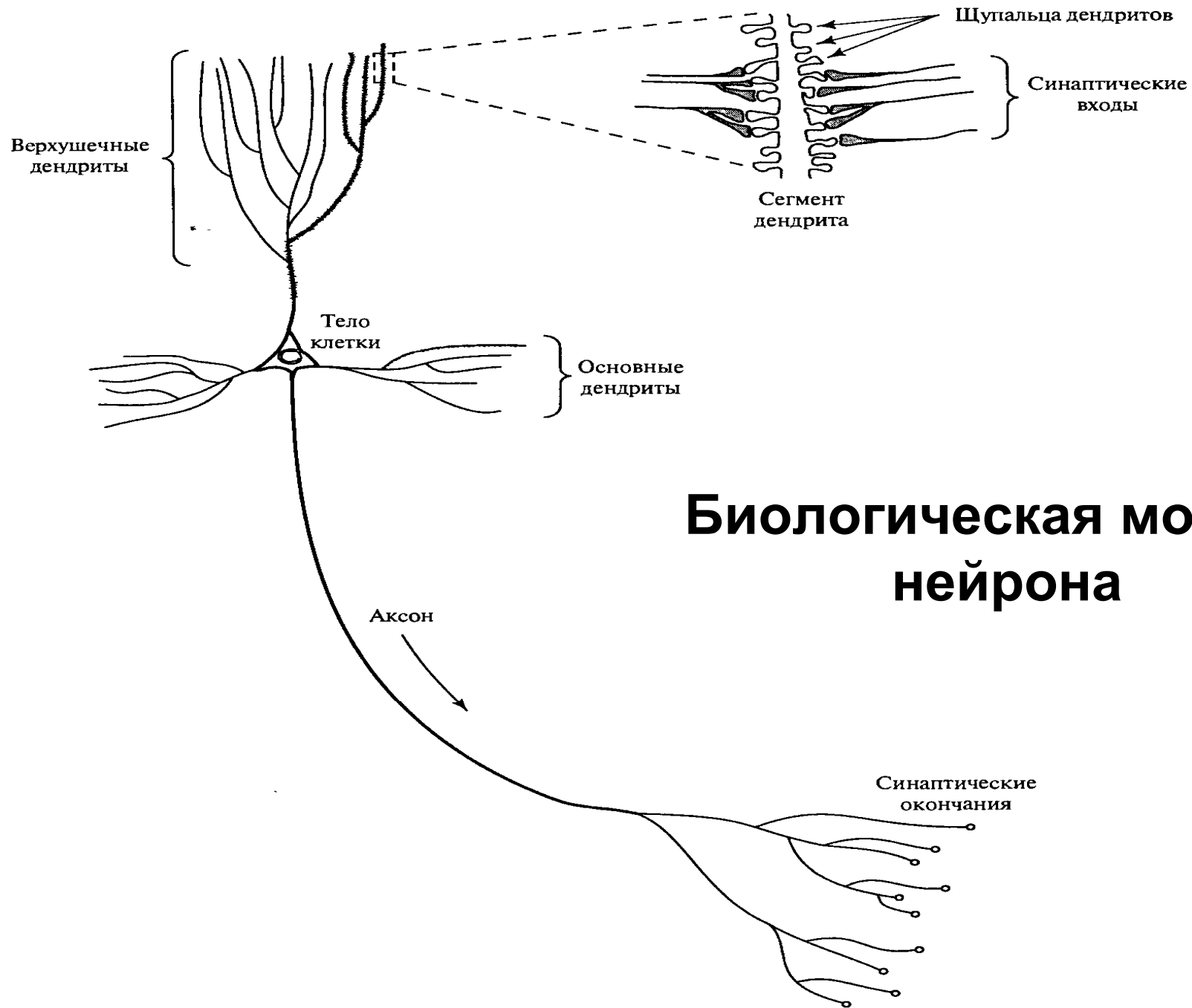


Рис. 1.2. Пирамидальная клетка

Работа нейрона

- Сигналы от других нейронов поступают на синаптические входы
- На щупальцах дендрита изменяются потенциалы
- В теле нейрона весь потенциал суммируется и добавляется еще собственный потенциал нейрона (порог)
- Если суммарный потенциал превышает некоторое значение (обычно это 0), то нейрон активизируется и посылает импульс (=1) на аксон, иначе нейрон остается неактивным (=0)
- Аксон разветвляется и образует связи с другими нейронами

Некоторые факты

- Существует около 50 типов нейронов
- Тело нейрона имеет размер ~100 микрон
- В коре головного мозга имеется ~100 миллиардов нейронов и ~60 000 миллиардов синапсов
- Нейрон может получать более 10 000 сигналов и передавать их на 1000 других клеток
- Скорость распространения нервного импульса в аксоне ~100 м/с, а в медной проволоке ~100 000 000 м/с
- Энергетические затраты мозга на выполнение одной операции в секунду $\sim 10^{-16}$ Дж, а самого экономичного компьютера $> 10^{-6}$ Дж.

Структурная организация уровней мозга

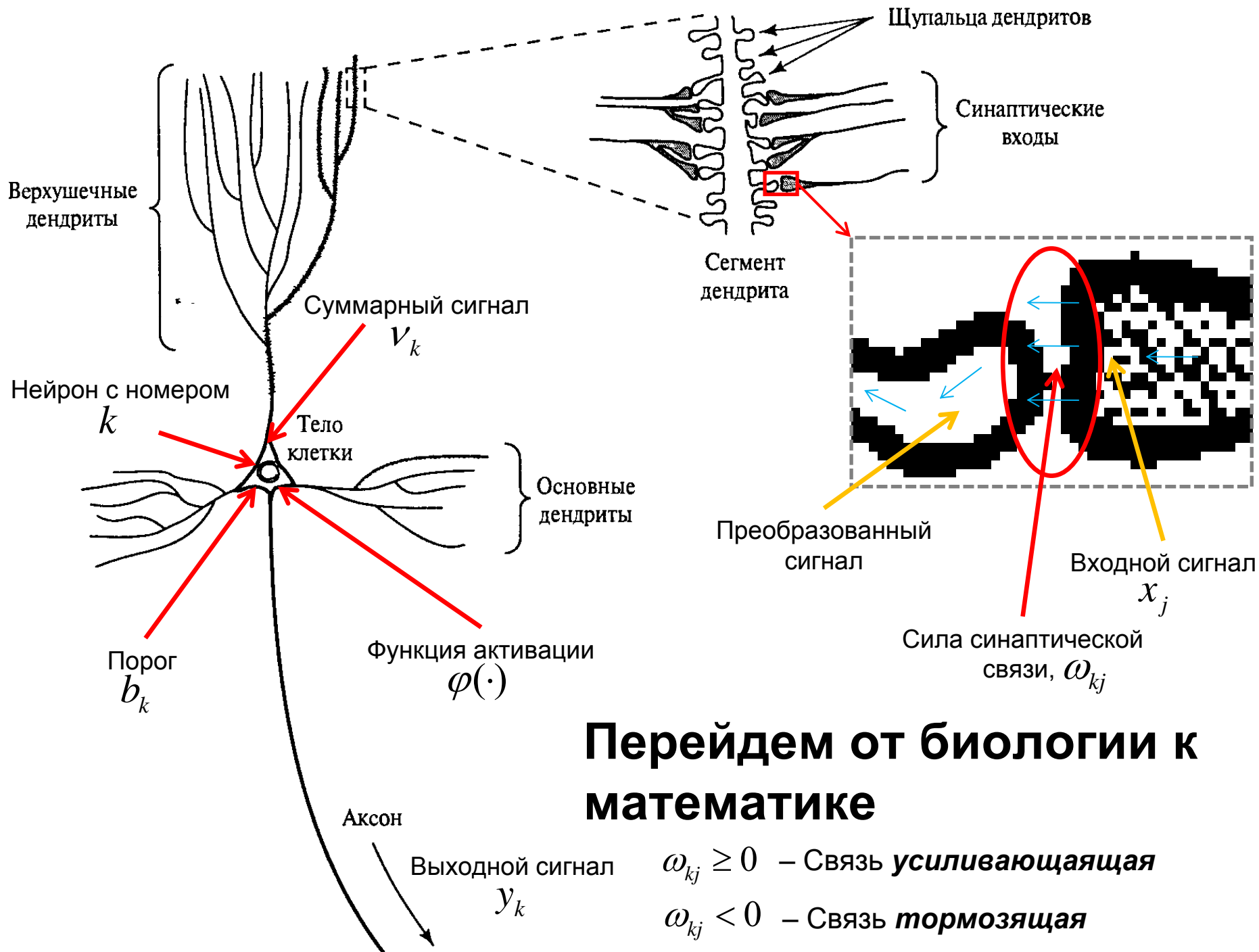


- *Нейронный микроконтур* – набор синапсов, выполняющих определенную операцию
- *Дендритное дерево* – совокупность нейронных микроконтуров
- *Локальные контуры* (~1 мм) – состоят из нейронов с одинаковыми или сходными характеристиками (отдельные области мозга)
- *Межрегиональные контуры* – объединяют несколько различных областей мозга
- Межрегиональные контуры связываются друг с другом, образуя *центральную нервную систему*

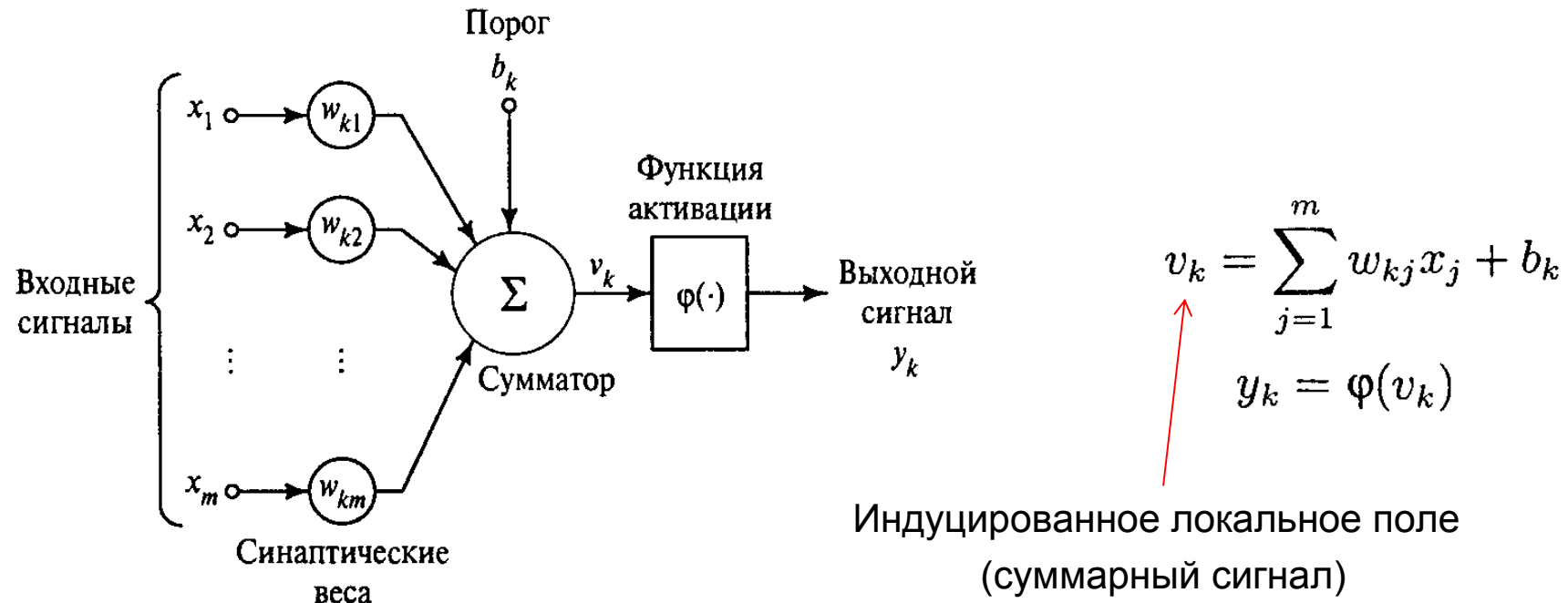
Структурная организация уровней мозга



- Описанные структурные уровни организации – уникальные характеристики мозга
- *Искусственные нейроны* сильно упрощены по сравнению с биологическими нейронами
- *Искусственные Нейронные сети* примитивны по сравнению с локальными и межрегиональными контурами мозга

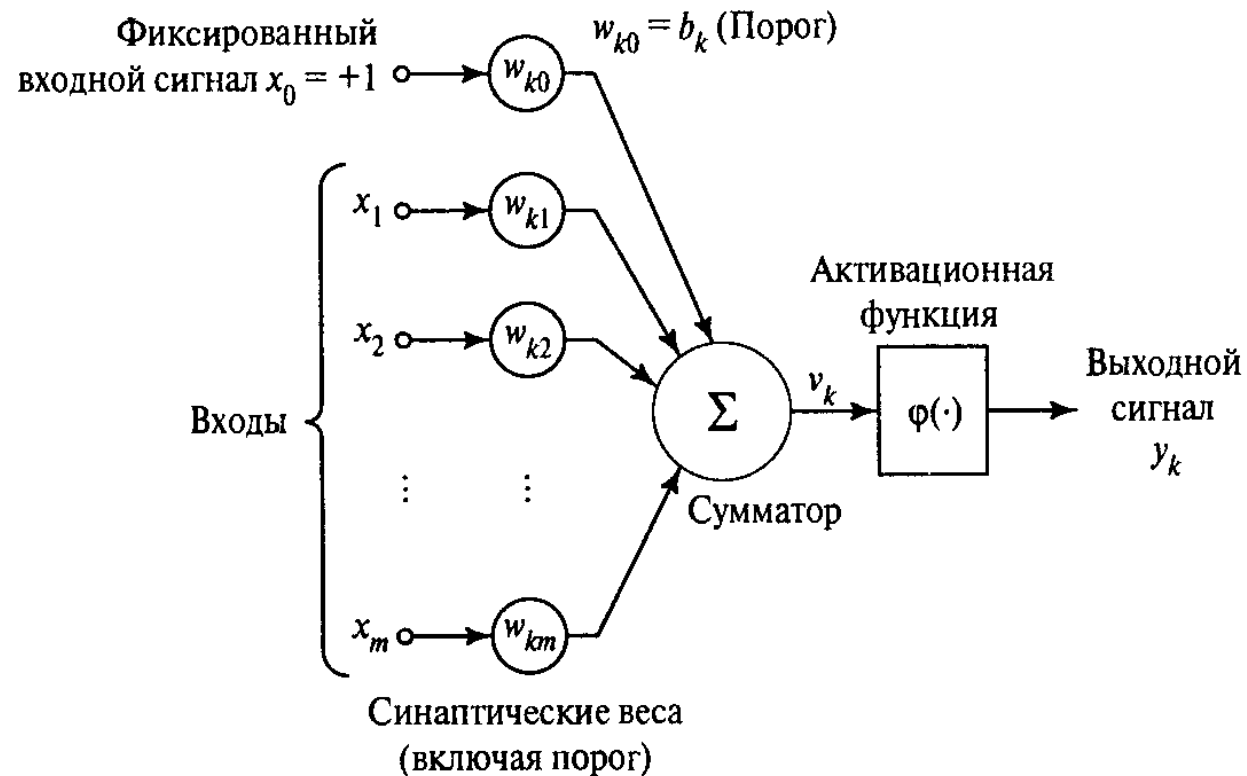


Математическая Модель Нейрона



- Чем больше порог b_k , тем легче активизировать нейрон
- Порог b_k можно считать фиксированным входным сигналом $x_0 = +1$ с синаптическим весом $w_{k0} = b_k$

Математическая Модель Нейрона



$$v_k = \sum_{j=0}^m w_{kj} x_j$$

$$y_k = \varphi(v_k)$$

■ Когда $y_k = \begin{cases} 1, & \text{если } v_k \geq 0; \\ 0, & \text{если } v_k < 0; \end{cases}$

– модель Мак-Каллока-Питца

Детерминистская Модель Нейрона

- Преобразование входного сигнала в выходной точно определено для всех значений входного сигнала
- По x_1, \dots, x_n однозначно восстанавливаем y
- Другими словами, имея входные сигналы – однозначно определяем выходной сигнал по формуле

$$y_k = \varphi(v_k)$$

Функции активации

- Выбор функций активации в SNN (путь):
 - Стартовое окно: *Анализ – Нейронные сети*
 - Выбираем: *Конструктор сетей*
 - Нажимаем: *ОК*
 - Открывается окно *Конструктора сетей*, выбираем вкладку: *Быстрый*
 - Нажимаем кнопку: *Правка*
 - Открывается окно *Редактора сети*, выбираем вкладку: *Слои*

Функции активации

Редактор сети: Таблица данных3

N	Б...	От...	Отн...	Архитектура	Производ...	Контр. про...	Тест. прои...	Ошибка о...	Конг
3		Y	0	МП 2:2-2:1:1	0,000000	0,000000	0,000000	0,000000	0,00

Слой	Элем...	Синаптичес...	Ф-ция акти..
1	2	Линейный	Линейная
2	2	Линейный	Гиперболич..
3	1	Линейный	Логистичес..

Синаптическая ф-ция: **Линейная**

Функция активации: **Логистическая**

Редактор моделей позволяет редактировать сеть и не записывается в макрос.

■ Синаптическая функция

Синаптическая ф-ция: **Линейная**

Функция активации: **Линейная**

- Линейная
- Радиальная
- ОРНС делитель

■ Функция активации

Синаптическая ф-ция: **Линейная**

Функция активации: **Тождественная**

- Тождественная
- Логистическая
- Гиперболическая
- Экспонента
- Софтмакс
- Сумма элементов
- Корень
- Синус
- Линейного роста
- Ступенчатая

Функции активации

Название	Функция, $\varphi(x)$	Значения
Линейная	x	$(-\text{inf}, +\text{inf})$
Ступенчатая (Хевисайда)	$\begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$	$[0, +1]$
Логистическая	$\frac{1}{1 + e^{-x}}$	$(0, +1)$
Гиперболический тангенс	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$(-1, +1)$
Экспонента	e^{-x}	$(0, +\text{inf})$

Функции активации

Софтмакс	$\frac{e^x}{\sum_i e^{x_i}}$	(0,+1)
Единичная сумма	$\frac{x}{\sum_i x_i}$	(0,+1)
Квадратный корень	\sqrt{x}	(0, +inf)
Синус	$\sin(x)$	[0,+1]
Пилообразная	$\begin{cases} -1, x \leq -1 \\ x, -1 < x < 1 \\ +1, x \geq 1 \end{cases}$	[-1,+1]

Вероятностная Модель Нейрона

- Обобщением детерминистской модели является вероятностная модель
- Нейрон находится в одном из двух состояний: +1 или -1

$$x = \begin{cases} +1, & \text{с вероятностью } P(v); \\ -1, & \text{с вероятностью } 1 - P(v). \end{cases}$$

Состояние нейрона

$$P(v) = \frac{1}{1 + \exp(-v/T)} \quad \text{– Вероятность активации нейрона}$$

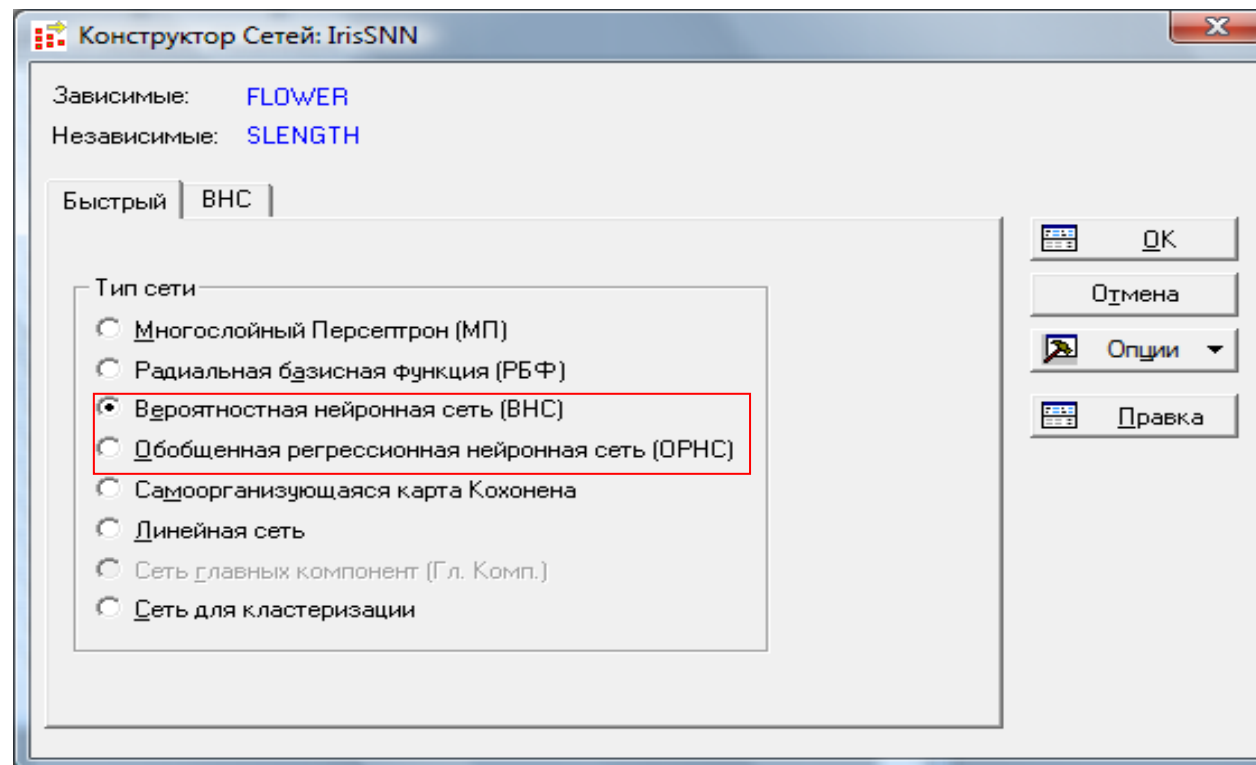
v – Индуцированное локальное поле нейрона

T – Аналог температуры, используемый для управления уровнем шума

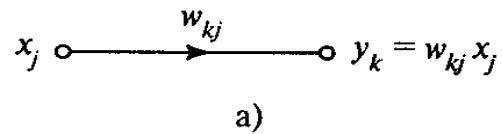
Вероятностная модель $\xrightarrow{T \rightarrow 0}$ Детерминистская модель Мак-Каллока-Питца

Вероятностные сети в SNN

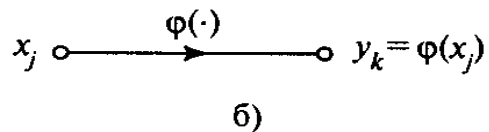
- Вероятностные нейронные сети основаны на вероятностной модели нейрона
- Вопрос: где они работают?
- В задачах классификации и регрессии



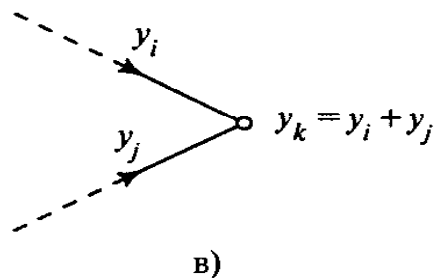
Нейронные сети как ориентированный граф



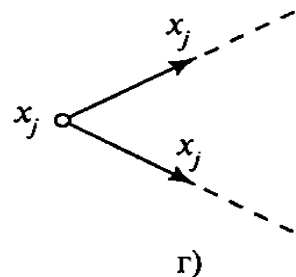
– Синаптическая связь (*линейная*)



– Активационная связь (*нелинейная*)



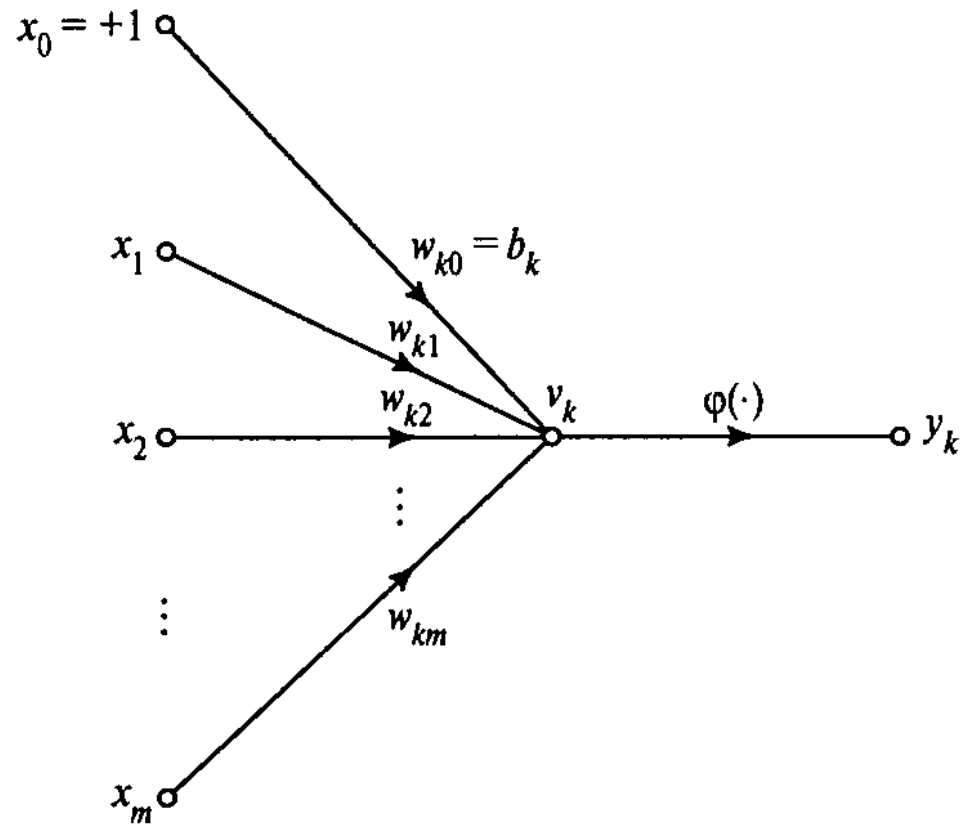
– Сигнал узла равен сумме входных сигналов (*синаптическая сходимость*)



– Сигнал узла передается по каждой исходящей связи (*синаптическая дивергенция*)

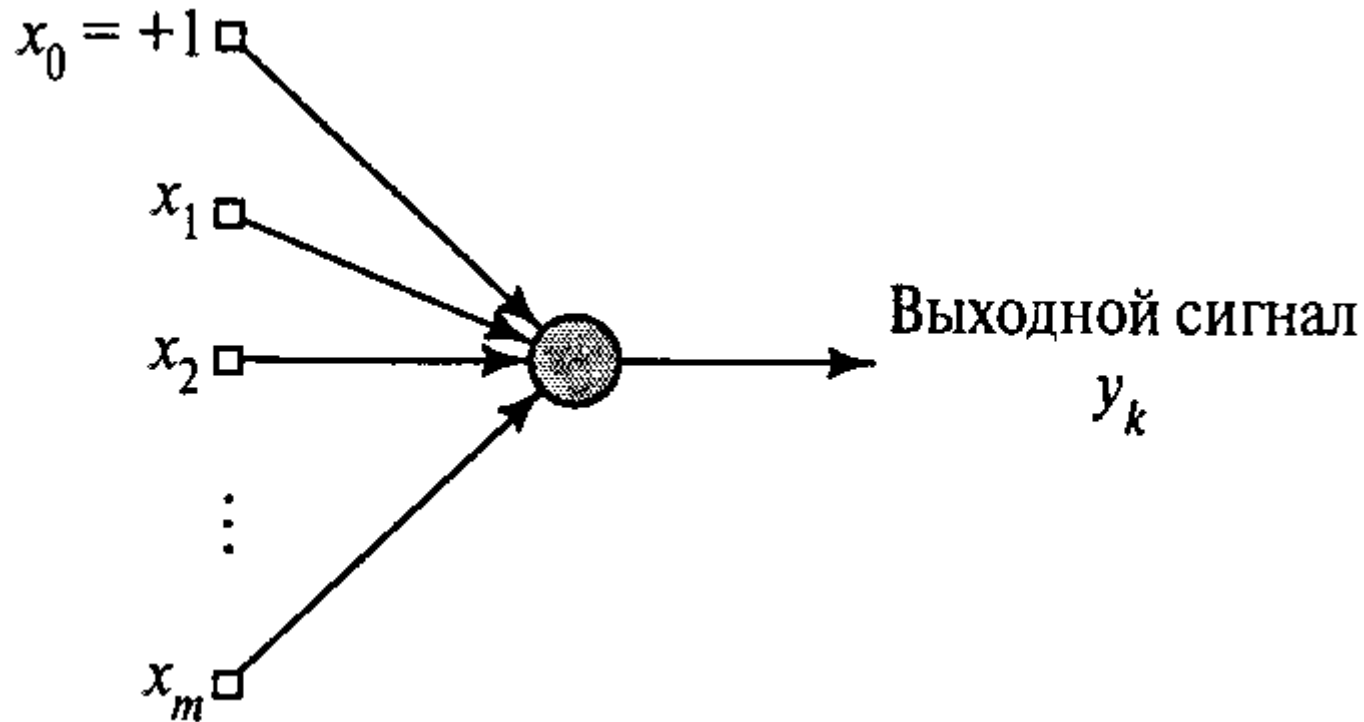
Рис. 1.9. Основные правила построения графов передачи сигналов

Нейронные сети как ориентированный граф



- **Полный ориентированный граф** описывает как прохождение сигнала между нейронами, так и передачу сигнала в самом нейроне

Нейронные сети как ориентированный граф



- **Частично полный (или структурный) ориентированный граф** описывает только прохождение сигнала между нейронами

Нейронные сети как ориентированный граф

Существует 3 способа графического представления нейронных сетей:

- **Блочная диаграмма**, описывающая функции нейронной сети
- **Граф прохождения сигнала**, обеспечивающий полное описание передачи сигнала
- **Структурный граф**, описывающий структуру нейронной сети

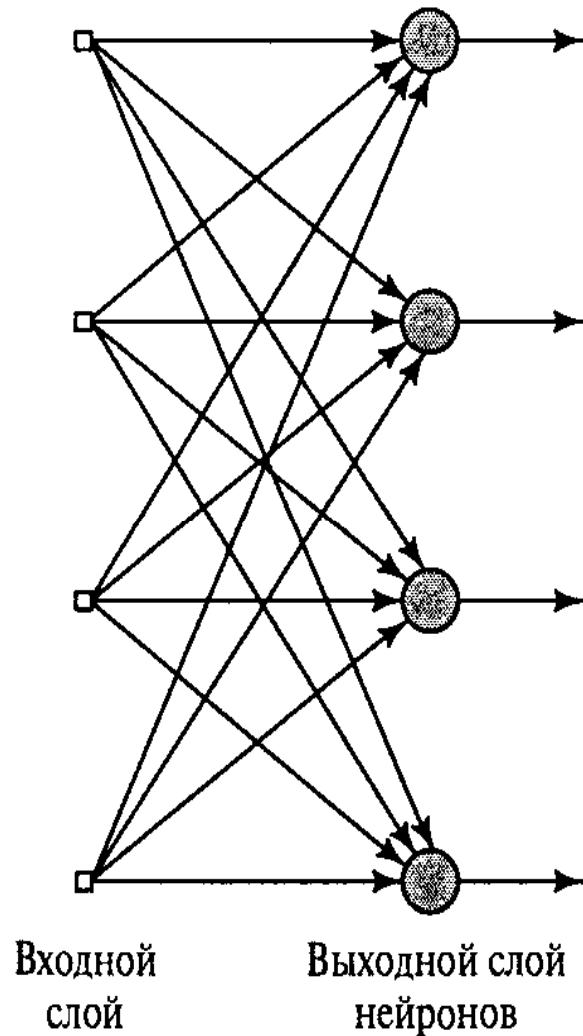
Архитектура сетей

Нейронная сеть – это совокупность нейронов, связанных между собой

- **Виды сетевых архитектур:**

- Однослойные сети прямого распространения
- Многослойные сети прямого распространения
- Рекуррентные сети (в SNN не реализованы)

Однослойная сеть



- Эта сеть состоит из двух слоев:
 - Входной
 - Выходной

- **Прямое распространение** – сигнал распространяется слева направо

- Данная сеть – **сеть 4-4**
(4 входных и 4 выходных нейрона)

Рис. 1.15. Сеть прямого распространения с одним слоем нейронов

Многослойная сеть

- **Полносвязная сеть** – все узлы каждого конкретного слоя соединены со всеми узлами смежных слоев
- Данная сеть – **сеть 10-4-2** (10 входных, 4 скрытых и 2 выходных нейрона)

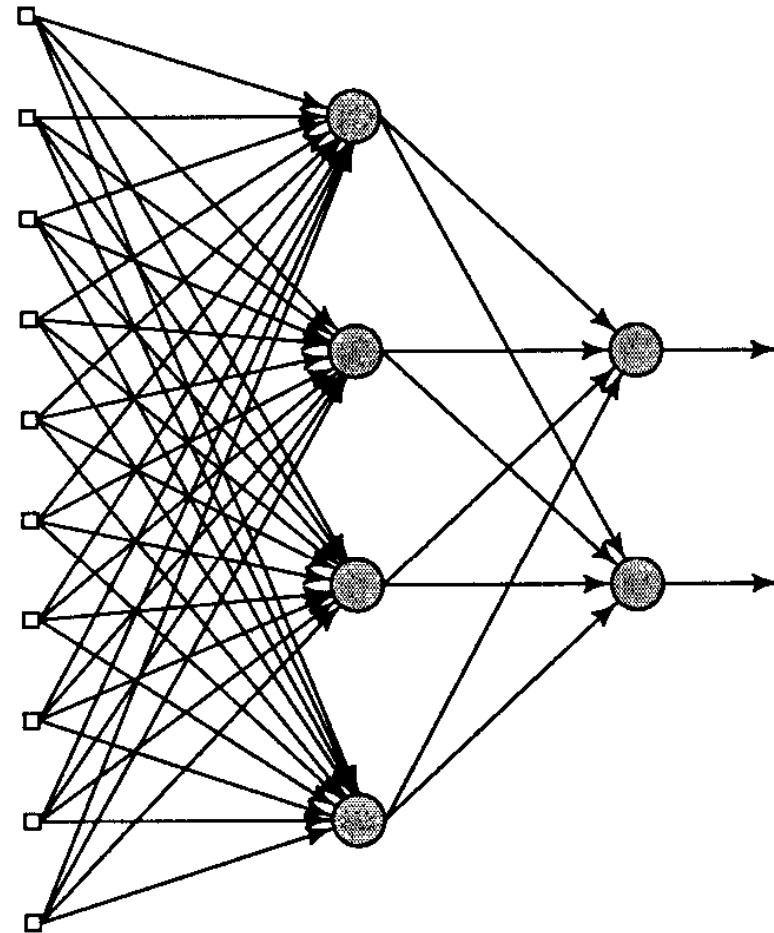


Рис. 1.16. Полносвязная сеть прямого распространения с одним скрытым и одним выходным слоем

Входной
слой

Слой скрытых
нейронов

Выходной слой
нейронов

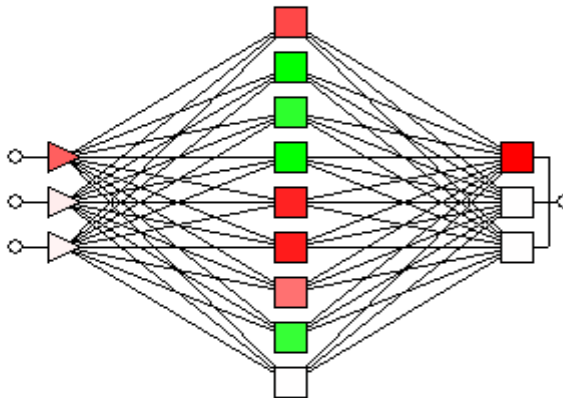
Архитектура сетей SNN

- В SNN все сети являются **ПОЛНОСВЯЗНЫМИ**

Параметры сети: IrisSNN

N	Архитектура	Производитель...	Контр. произво...	Тест. производ...	Ошибка обучен...	Кс
1	Линейная 2:2-3:1	0,815789	0,837838	0,837838	0,314889	0,;
2	Линейная 3:3-3:1	0,881579	0,837838	0,864865	0,292774	0,;
3	МП 4:4-10-3:1	0,973684	0,972973	0,972973	0,172480	0,;
4	МП 3:3-9-3:1	0,973684	1,000000	0,972973	0,178711	0,;
5	РБФ 4:4-19-3:1	0,986842	0,972973	0,945946	0,109944	0,;

Архитектура | Наблюдение пользователя



Отмена

Опции

Выбрать модели

Архитектура сети

Все сети

Выделить активацию цветом

Входное набл.: 1

Примеры

- В модели Мак-Каллока-Питца нейрон реализует логические операции (на выходе имеем 0 или 1)
- Посмотрим реализацию логических операций «И», «ИЛИ», «XOR» в нейронных сетях

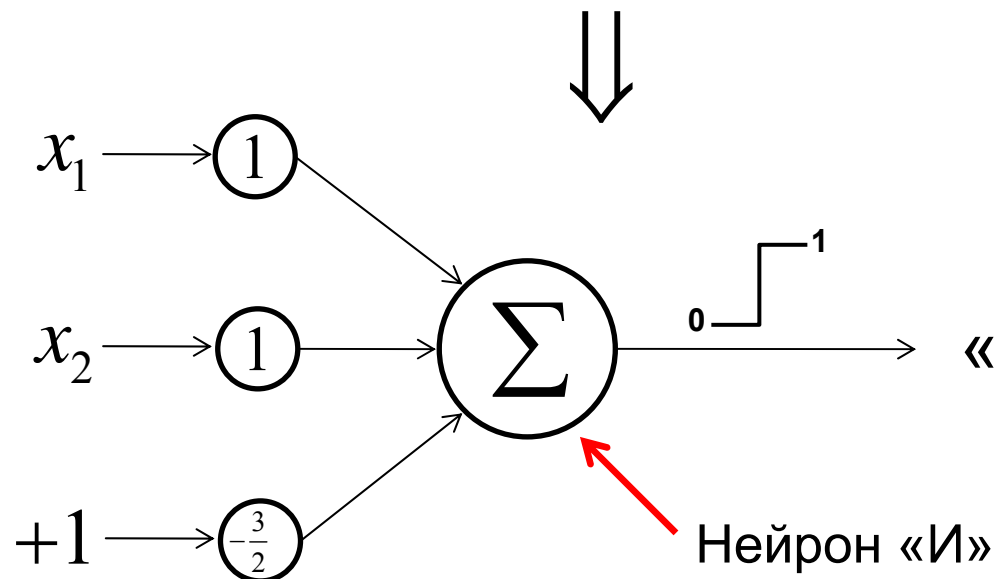
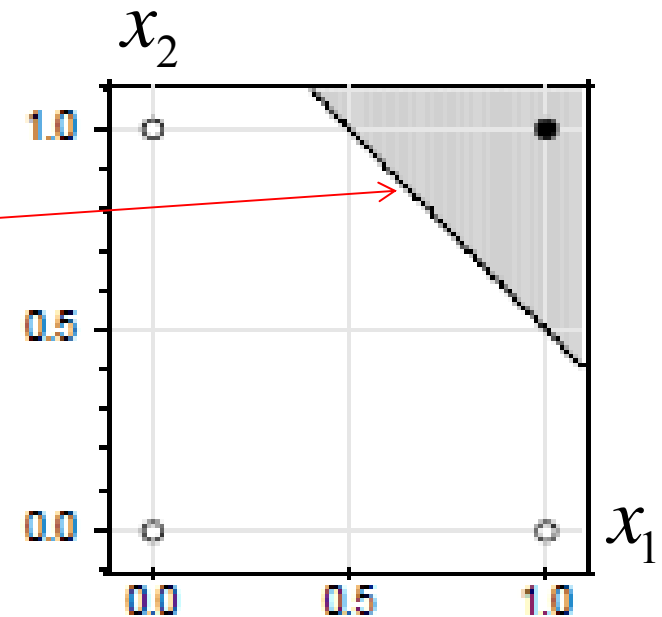
x_1	x_2	И	ИЛИ	XOR
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	1	1

Операция «И»

- Рассмотрим прямую:

$$x_1 + x_2 - \frac{3}{2} = 0$$

- Выше прямой – класс «1»
- Ниже прямой – класс «0»



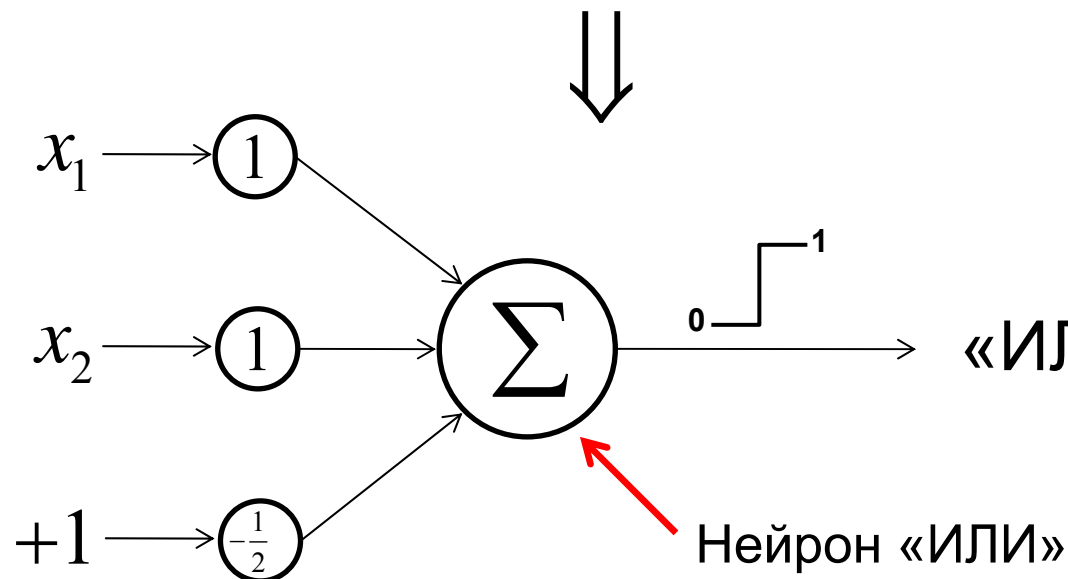
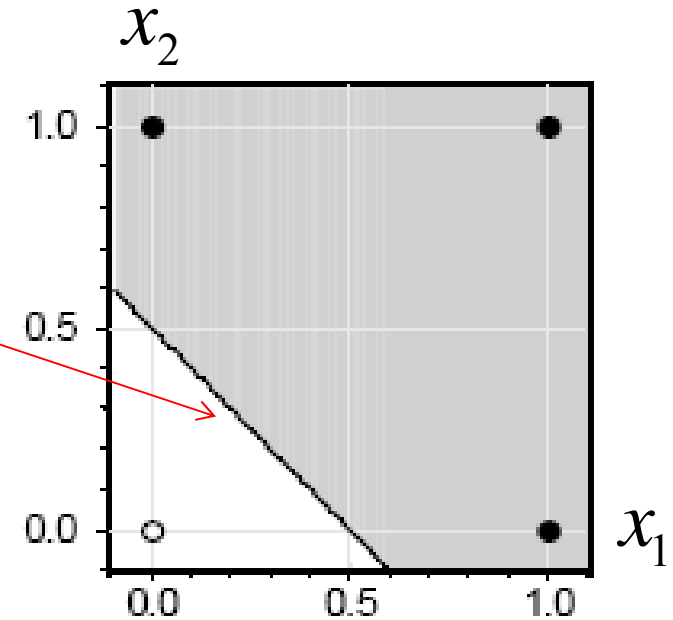
«И» - реализуется одним нейроном

Операция «ИЛИ»

- Рассмотрим прямую:

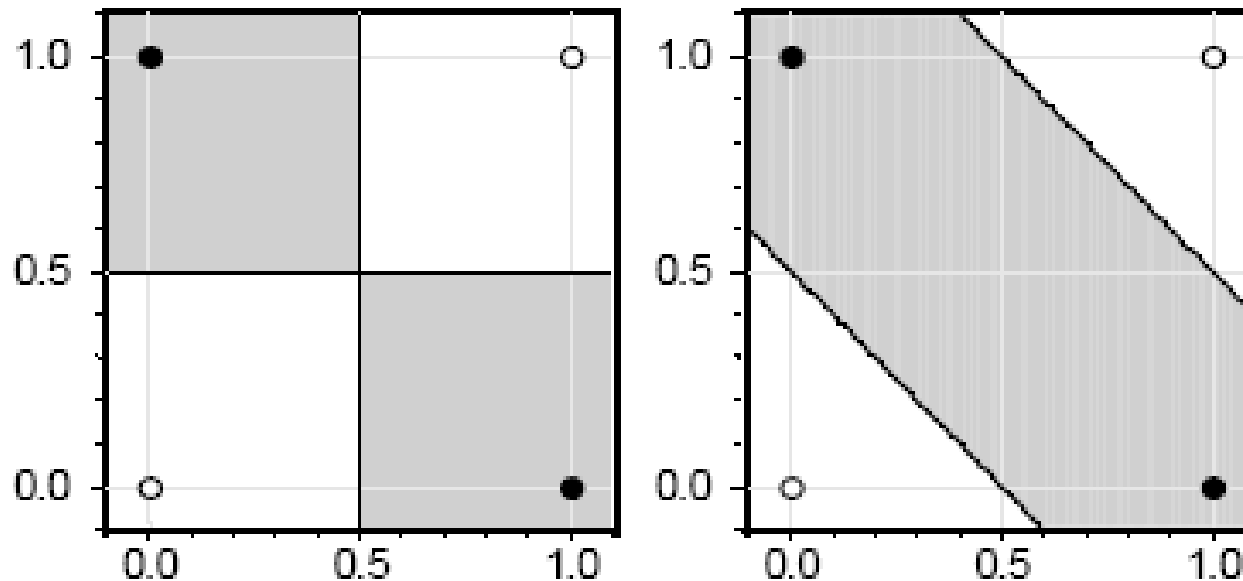
$$x_1 + x_2 - \frac{1}{2} = 0$$

- Выше прямой – класс «1»
- Ниже прямой – класс «0»



«ИЛИ» - реализуется одним нейроном

Операция «XOR»



- **Нельзя** построить прямую, которая бы разделила точки на два класса.
- Что же делать в таком случае?

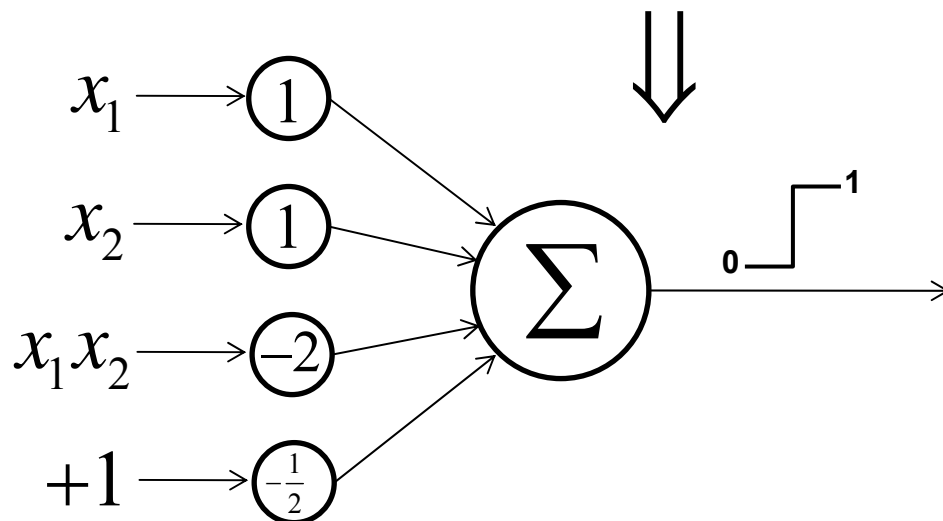
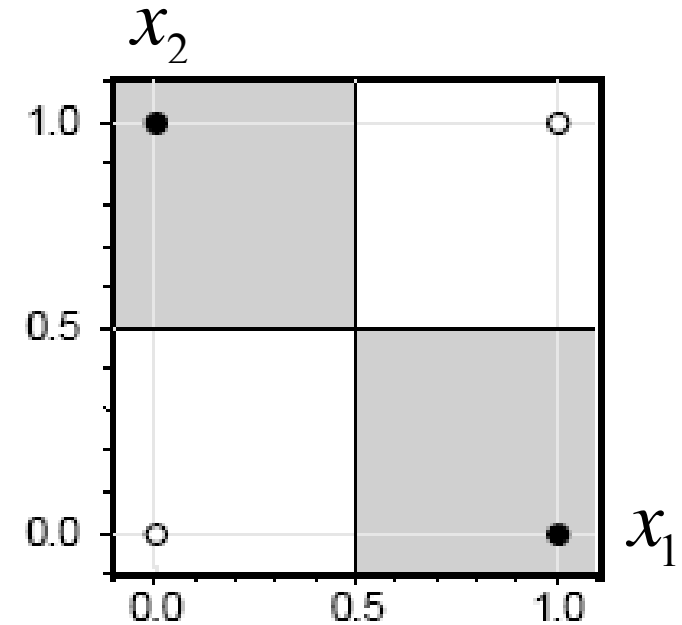
Операция «XOR», подход 1

- Рассмотрим множество точек:

$$x_1 + x_2 - 2x_1x_2 - \frac{1}{2} = 0$$

(две прямые образующие «+»)

- Темная область – класс «1»
- Светлая область – класс «0»



«XOR» - реализуется одним нейроном с добавлением нелинейного признака

Операция «XOR», подход 1

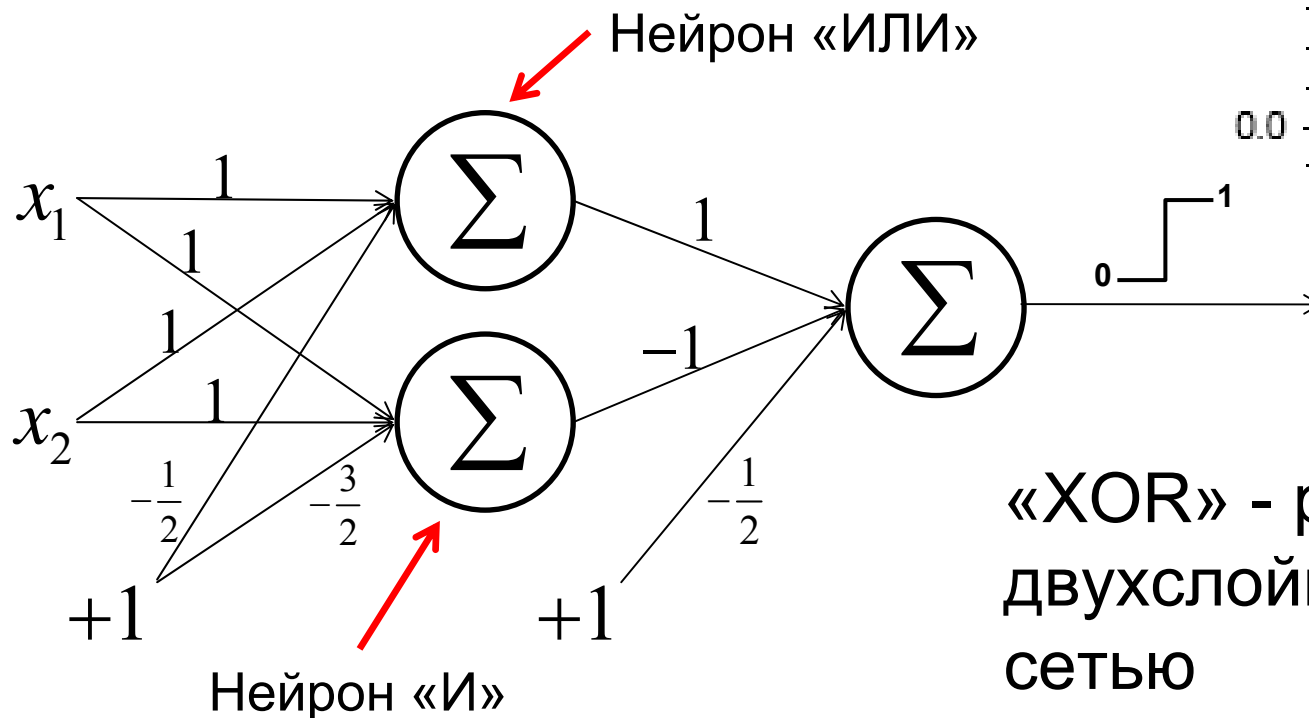
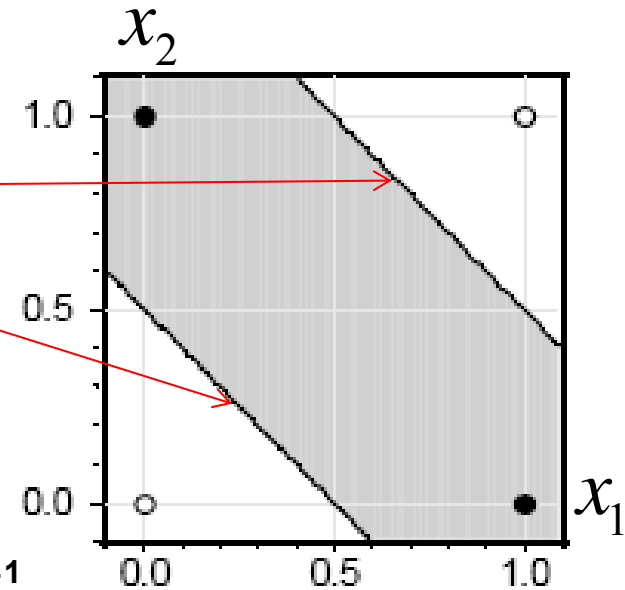
■ Проблема

- подбор нужных нелинейных преобразований – нетривиальная задача.
- до сих пор эта задача остаётся нерешённой для общего случая.

Операция «XOR», подход 2

- Использовать композицию нейронов:

$$\text{ИЛИ} - \text{И} - \frac{1}{2} = 0$$



«XOR» - реализуется двухслойной нейронной сетью

Вопрос: любую ли функцию можно представить с помощью нейронных сетей?

- **Теорема. (Колмогоров, 1957).** Любая непрерывная функция n аргументов на единичном кубе $[0, 1]^n$ представима в виде суперпозиции непрерывных функций одного аргумента и операции «+»:

$$f(x^1, x^2, \dots, x^n) = \sum_{k=1}^{2n+1} h_k \left(\sum_{i=1}^n \varphi_{ik}(x^i) \right)$$

Функция активации
скрытого нейрона

Скрытый слой
нейронов

Входной слой
нейронов

Функции
синаптической
связи

Двухслойная нейронная
сеть описывает почти
любую функцию

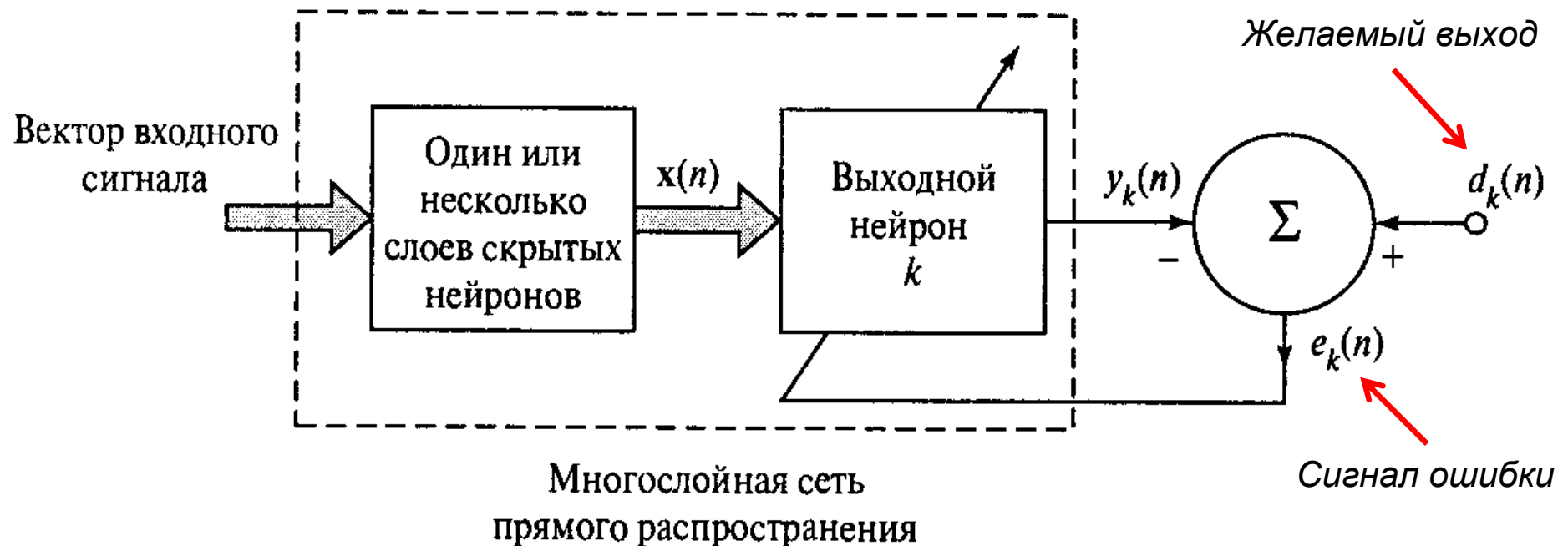
Принципы обучения нейросетей

- Самое важное свойство нейронных сетей – способность ***обучаться***
- Обучение происходит интерактивно
- При этом корректируются параметры сети: синаптических веса и пороги

Основные шаги обучения

1. В нейронную сеть поступают *стимулы* из внешней среды
2. В результате этого *изменяются* свободные параметры нейронной сети
3. После изменения внутренней структуры нейронная сеть *отвечает* на возбуждение уже иным образом

Обучение на основе коррекции ошибок

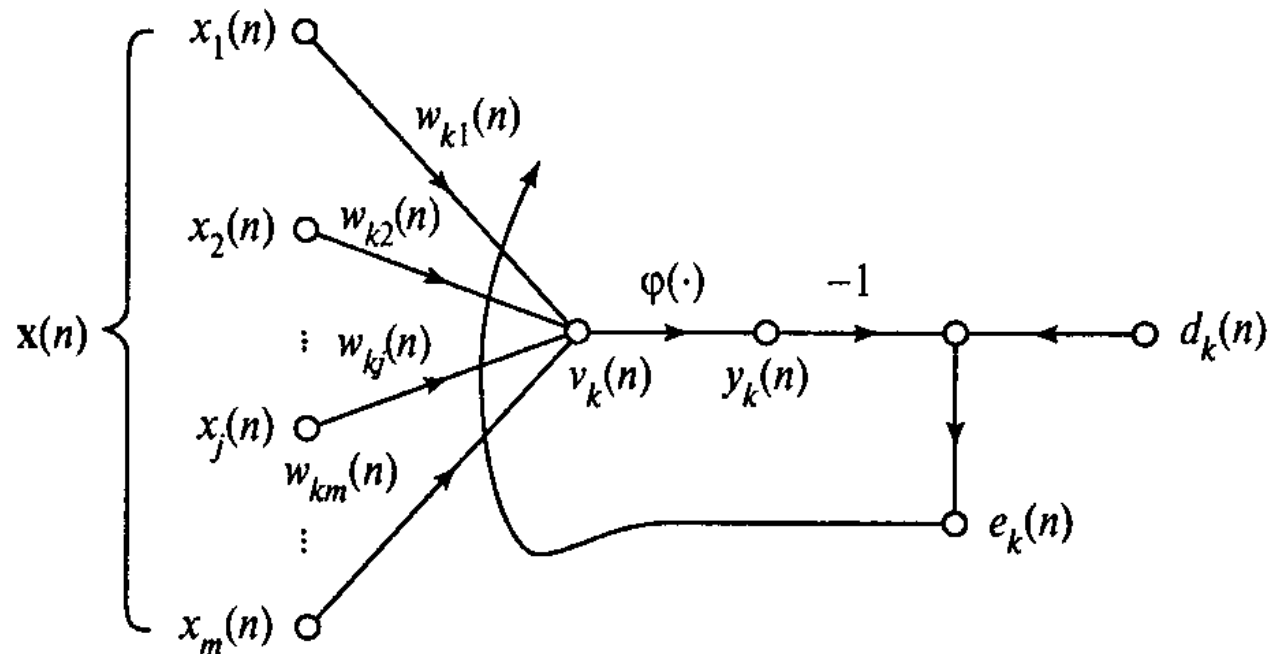


$$e_k(n) = d_k(n) - y_k(n)$$

Функция стоимости –
$$\mathbf{E}(n) = \frac{1}{2} e_k^2(n) \rightarrow \min$$

n – номер шага итеративного процесса настройки синаптических весов нейрона k

Обучение на основе коррекции ошибок



$E(n)$ минимизируется по **дельта-правилу (правило Видроу-Хоффа)**

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

η — некоторая положительная константа, определяющая *скорость обучения*

Обучение на основе памяти

- Рассмотрим задачу классификации (распознавание образов)

- Имеется обучающая выборка $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$

Входной вектор *Желаемый выход (класс)*

- Хотим классифицировать новый вектор \mathbf{x}_{test}

- \mathbf{x}_{test} относим к тому классу, которому принадлежат ближайšie к \mathbf{x}_{test} точки из обучающей выборки

Обучение на основе памяти

Необходимо выбрать:

- Критерий определения окрестности вектора \mathbf{x}_{test}
- Правило обучения, применяемое к обучающим векторам из окрестности вектора \mathbf{x}_{test}

Все алгоритмы обучения на основе памяти отличаются только реализацией этих двух этапов

Обучение на основе памяти

- Алгоритм *ближайшего соседа*

- Окрестность – определяется только ближайший к \mathbf{x}_{test} обучающий вектор $\mathbf{x}'_N \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$

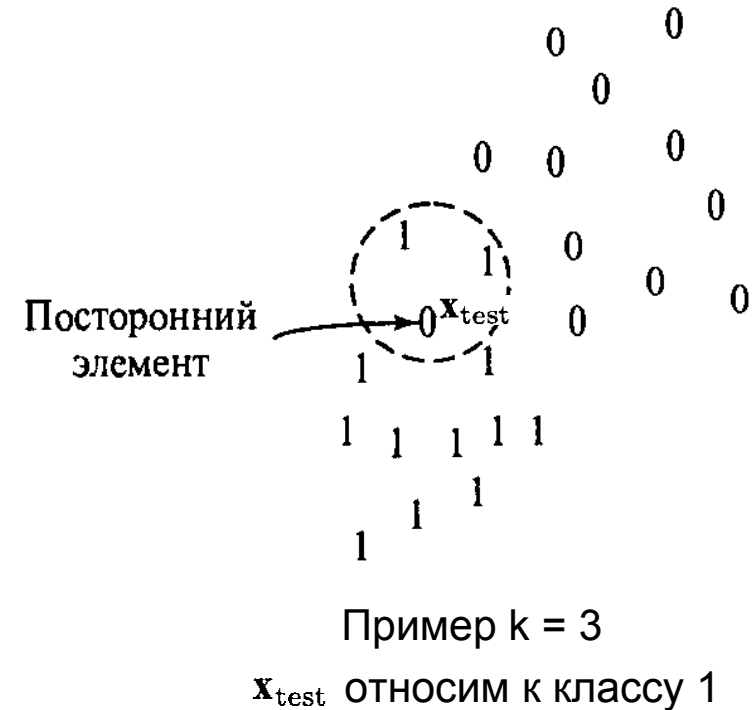
$$\min_i d(\mathbf{x}_i, \mathbf{x}_{\text{test}}) = d(\mathbf{x}'_N, \mathbf{x}_{\text{test}})$$

- Правило обучения – \mathbf{x}_{test} относим к тому классу, которому принадлежит ближайший обучающий вектор \mathbf{x}'_N

Обучение на основе памяти

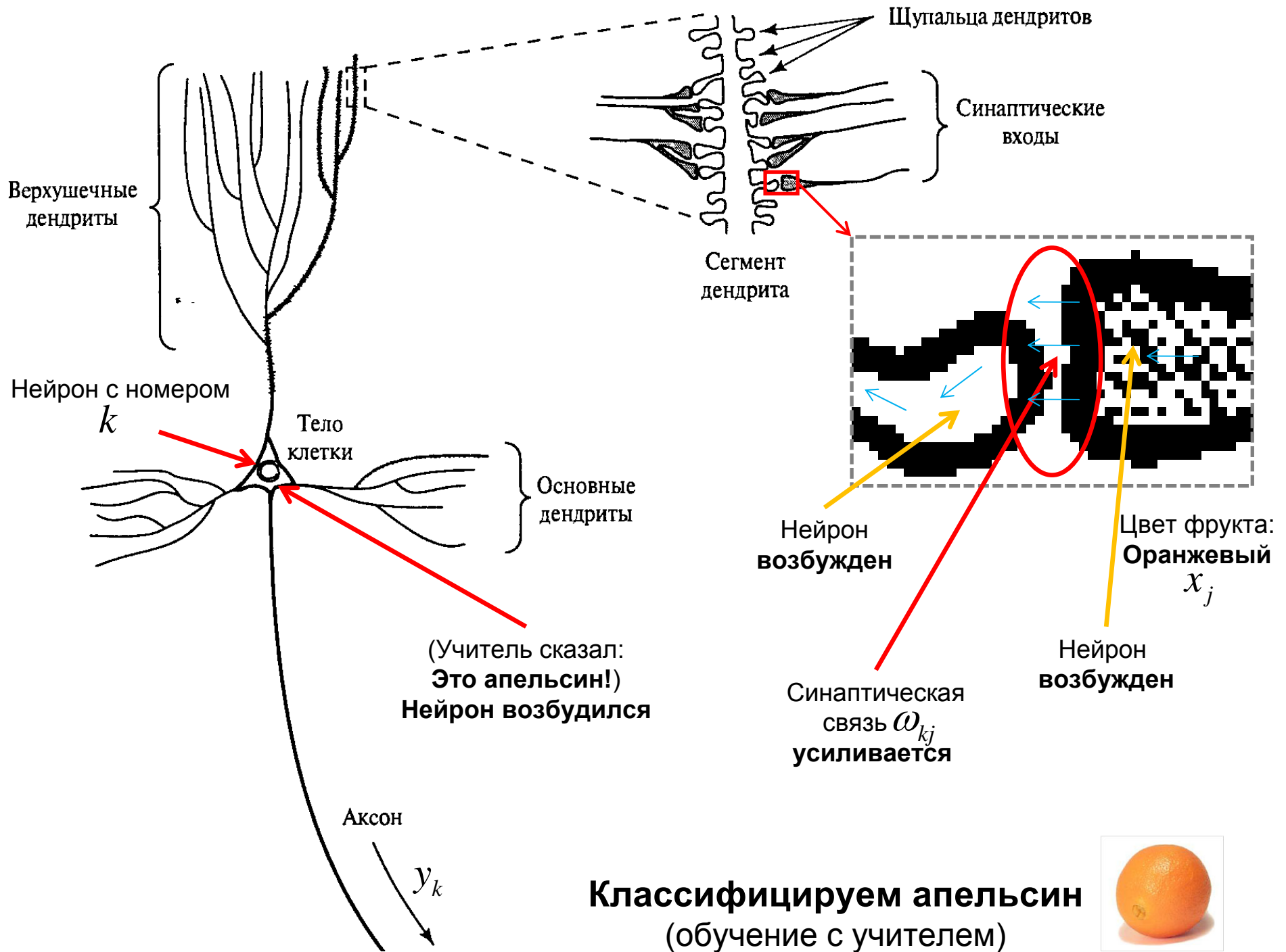
■ Алгоритм *k*-ближайших соседей

- Окрестность – определяем k ближайших к \mathbf{x}_{test} обучающих векторов
- Правило обучения – \mathbf{x}_{test} относим к тому классу, которому принадлежит большинство обучающих векторов из окрестности



Обучение Хебба

- Если два нейрона по обе стороны синапса **синхронно** возбуждаются – синаптическая связь **усиливается**
- Если два нейрона по обе стороны синапса **асинхронно** возбуждаются – синаптическая связь **ослабляется или отмирает**



Обучение Хебба

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

$$w_{kj}(n + 1) = w_{kj}(n) + \Delta w_{kj}(n)$$

η — некоторая положительная константа, определяющая *скорость обучения*

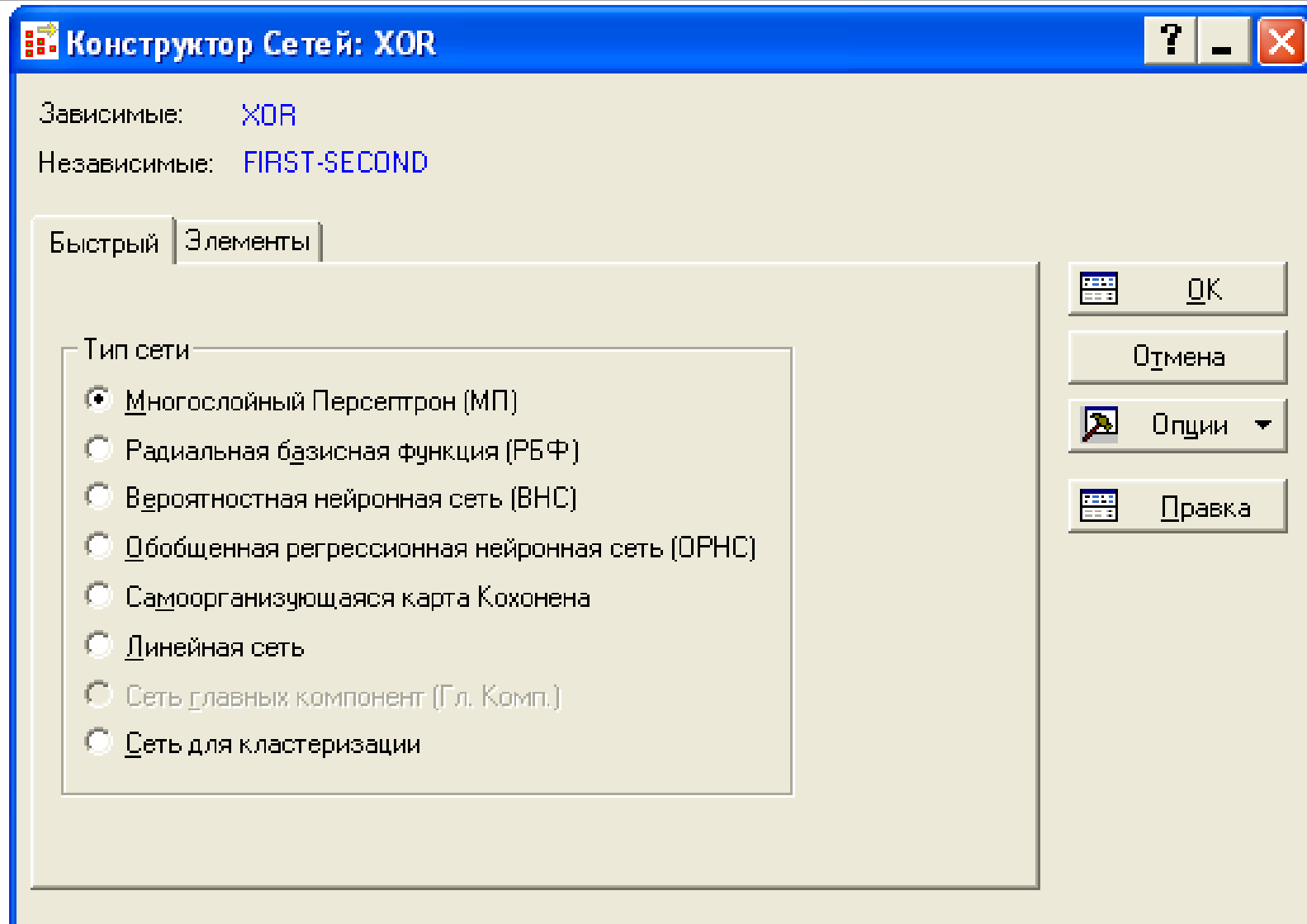
- некоторая положительная константа, определяющая скорость обучения

n — номер шага итеративного процесса (или номер входного вектора) настройки синаптических весов нейрона k

Задачи обучения

- Распознавание образов
- Аппроксимация функций
- Фильтрация
- Интерполяция
- Прогнозирование (Экстраполяция)
- ...

Виды сетей в SNN



Однослойный Персептрон (Линейная сеть)

Задача адаптивной фильтрации

- Функция активации – *линейная*

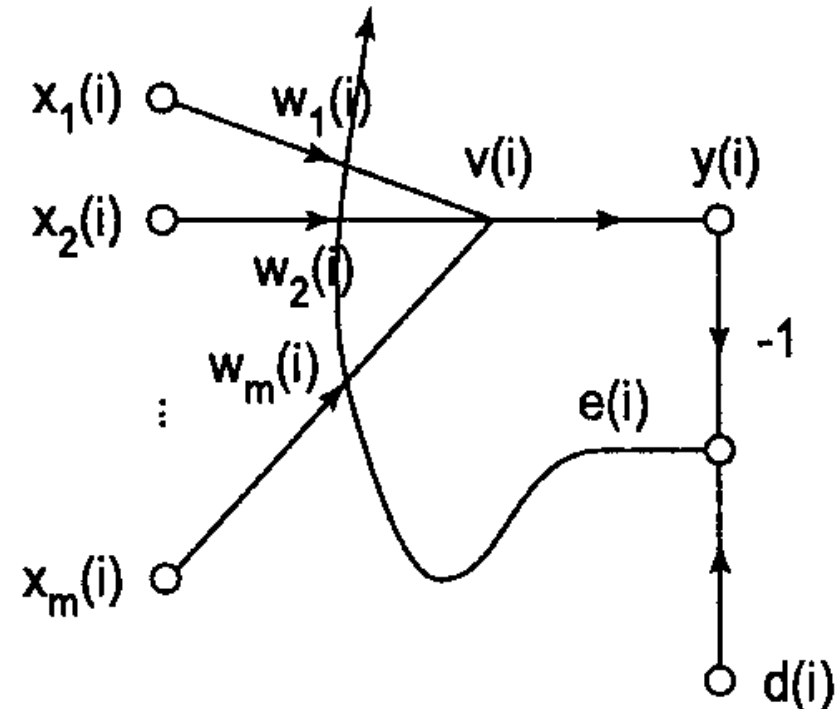
$$y(i) = v(i) = \sum_{k=1}^m w_k(i)x_k(i)$$

- Работа фильтра:

- Процесс фильтрации – вычисление $y(i)$ и $e(i)$
- Процесс адаптации – перенастройка весов

$$\mathbf{w}(i) = [w_1(i), w_2(i), \dots, w_m(i)]^T$$

$$E(\mathbf{w}) \rightarrow \min$$



Адаптивный фильтр

Методы безусловной оптимизации

- Формулировка: $E(\mathbf{w}) \rightarrow \min$

- Необходимое условие оптимальности:

$$\nabla E(\mathbf{w}^*) = 0.$$

$\nabla E(\mathbf{w})$ — вектор градиента функции стоимости

$$\nabla E(\mathbf{w}) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_m} \right]^T$$

- Последовательный спуск:

$$E(\mathbf{w}(n+1)) < E(\mathbf{w}(n))$$

Метод градиентного спуска

- Функция стоимости зависит сразу от всей выборки!

- Градиент: $\mathbf{g} = \nabla E(\mathbf{w})$

- Алгоритм градиентного спуска:

$$\mathbf{w}(n + 1) = \mathbf{w}(n) - \eta \mathbf{g}(n)$$

η — положительная константа, называемая *параметром скорости обучения*

$$\begin{aligned} \mathbf{E}(\mathbf{w}(n + 1)) &= \mathbf{E}(\mathbf{w}(n)) + \mathbf{g}^T(n) \Delta \mathbf{w}(n) \\ &= \mathbf{E}(\mathbf{w}(n)) - \eta \|\mathbf{g}(n)\|^2 \end{aligned}$$

Метод стохастического градиентного спуска



- Аналогичен градиентному спуску, **но**:
 - Функция стоимости зависит **только от одного** входного объекта!
 - Объекты перебираются в случайном порядке по одному
 - Для каждого входного объекта делается градиентный шаг
 - Сразу обновляется вектор весов

$$\mathbf{w}(n + 1) = \mathbf{w}(n) - \eta \mathbf{g}(n)$$

Задача адаптивной фильтрации

- Функция стоимости: $\mathbf{E}(\mathbf{w}) = 1/2e^2(n)$
- Сигнал ошибки: $\mathbf{e}(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n)$
- Пользуемся методом стохастического градиентного спуска:

$$\frac{\partial \mathbf{E}(\mathbf{w})}{\partial \mathbf{w}} = e(n) \frac{\partial e(n)}{\partial \mathbf{w}} = -\mathbf{x}(n)e(n)$$

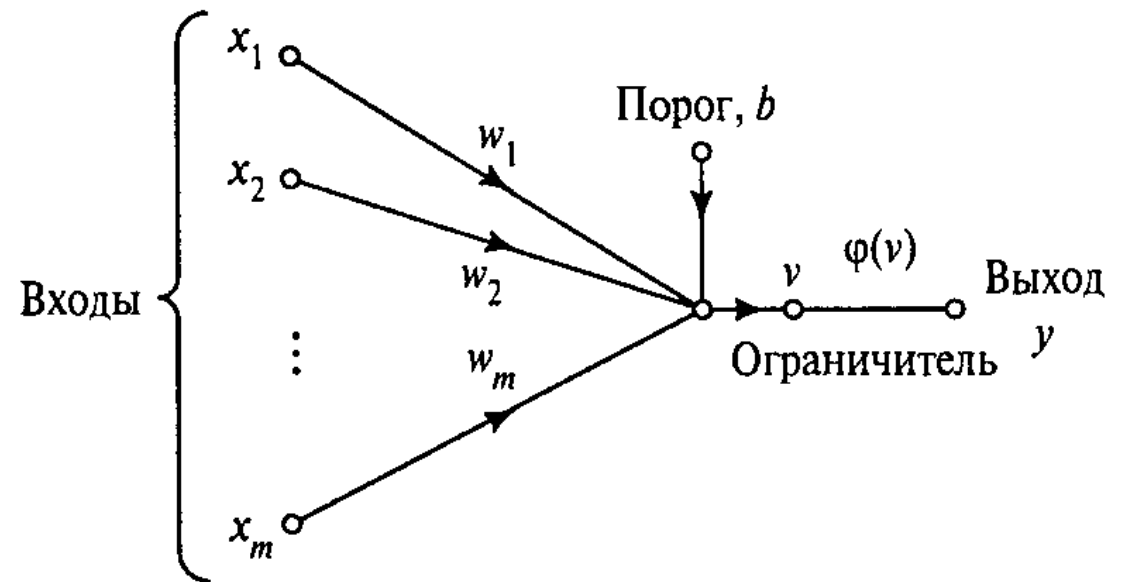
- Получаем **дельта-правило**:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta \mathbf{x}(n)e(n)$$

- Это возможно только из-за линейной функции активации

Персептрон

- Строится для модели нейрона Мак-Каллока-Питца



- Нелинейная функция активации:

$$\varphi(v) = \text{sgn}(v) = \begin{cases} +1, & \text{если } v > 0, \\ -1, & \text{если } v < 0. \end{cases}$$

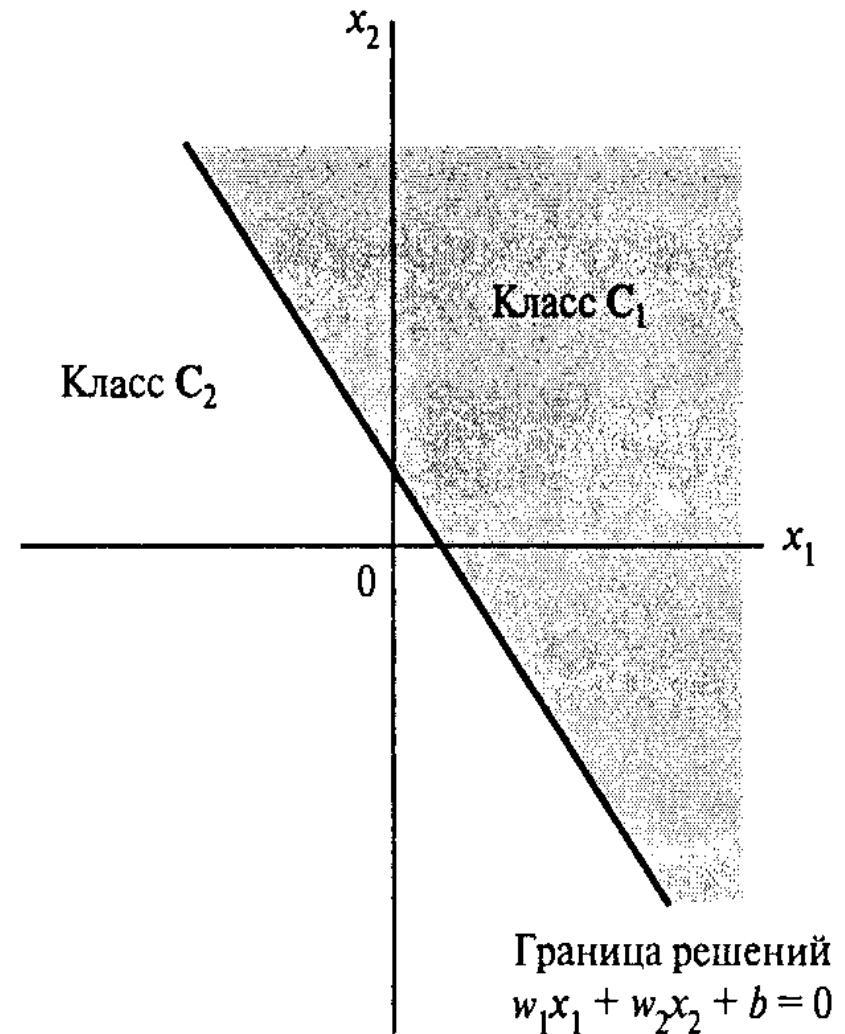
Персептрон

- Суммарный входной сигнал:

$$v = \sum_{i=1}^m w_i x_i + b.$$

- Уравнение разделяющей гиперплоскости:

$$\sum_{i=1}^m w_i x_i + b = 0$$



Персептрон

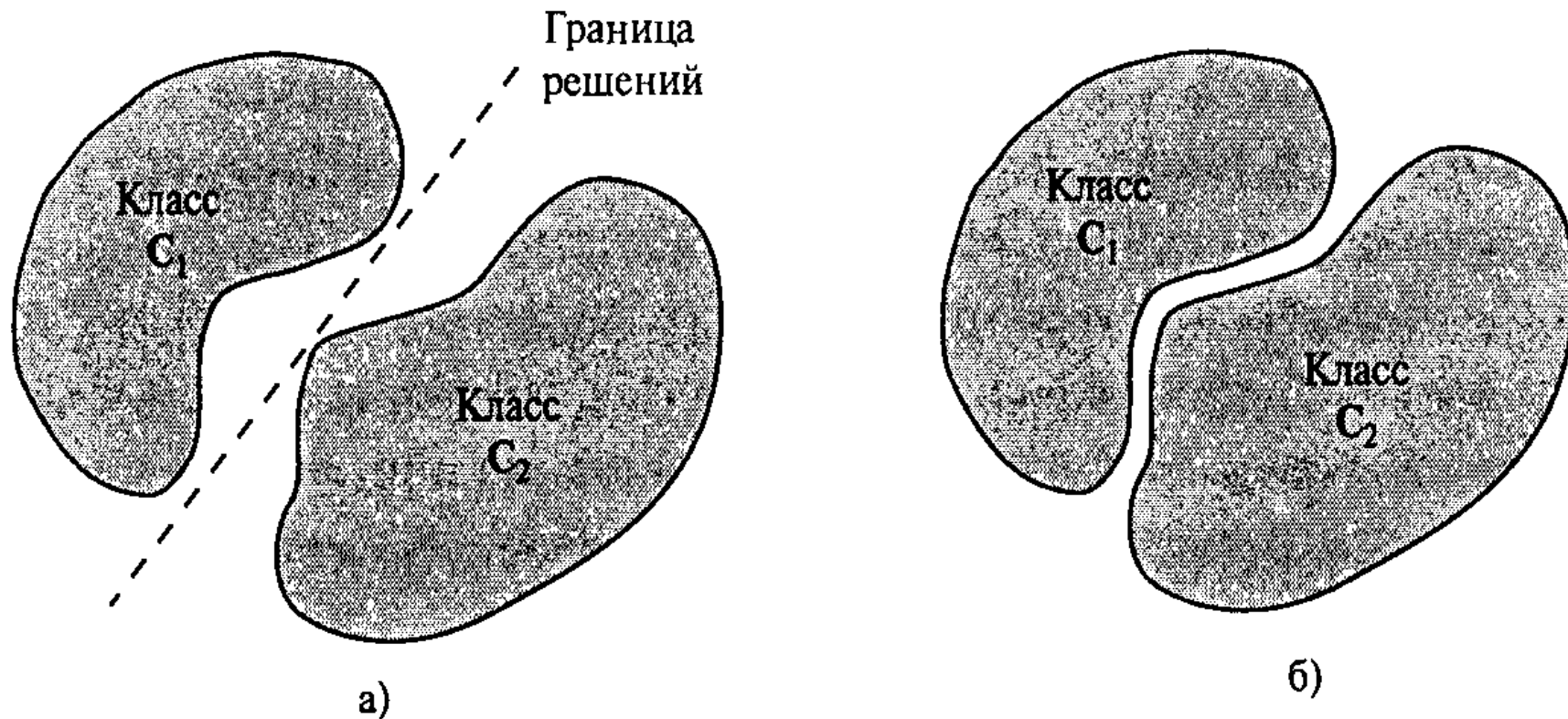
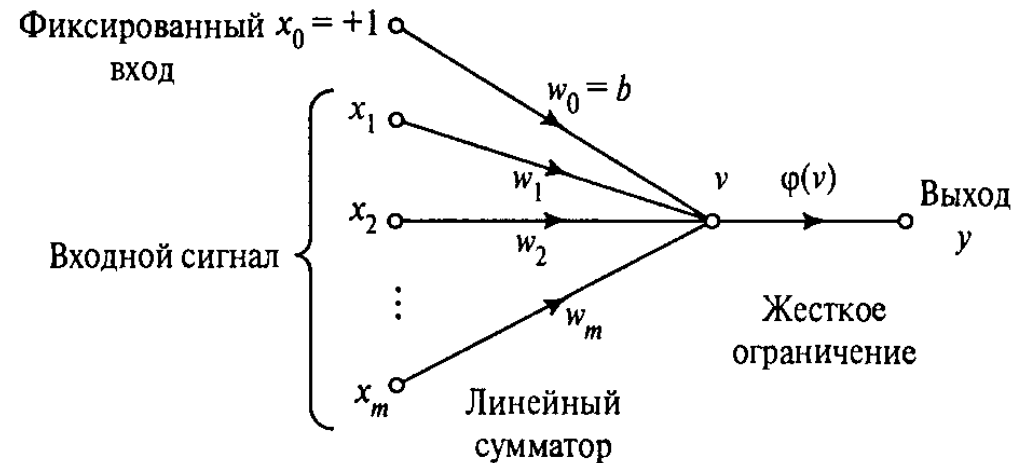
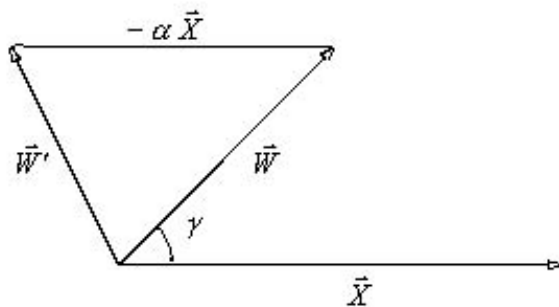
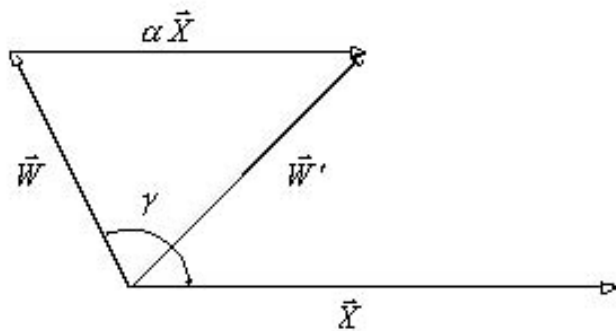


Рис. 3.9. Пара линейно-разделимых (а) и нелинейно-разделимых образов (б)

- Чтобы персептрон функционировал корректно, два класса, C_1 и C_2 , должны быть **линейно-разделимыми**

Теорема о сходимости персептрона



$$\mathbf{x}(n) = [+1, x_1(n), x_2(n), \dots, x_m(n)]^T$$

$$\mathbf{w}(n) = [b(n), w_1(n), w_2(n), \dots, w_m(n)]^T$$

$$y(n) = \text{sgn}(\mathbf{w}^T(n)\mathbf{x}(n))$$

Теорема сходимости персептрона

- Правило обучения персептрона (коррекция ошибок)

$$\mathbf{w}(n + 1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$

- Теорема сходимости персептрона:

Если выборка линейно разделима то алгоритм обучения сходится за конечное число шагов, т.е для некоторого n_0

$$\mathbf{w}(n_0) = \mathbf{w}(n_0 + 1) = \mathbf{w}(n_0 + 2) = \dots$$

Алгоритм обучения персептрона

1. Инициализация: $\mathbf{w}(0)=\mathbf{0}$.

2. Вычисление фактического ответа:

$$y(n) = \text{sgn}(\mathbf{w}^T(n)\mathbf{x}(n)).$$

3. Адаптация вектора весов:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n),$$

где

$$d(n) = \begin{cases} +1, & \text{если } \mathbf{x}(n) \in \mathbf{C}_1, \\ -1, & \text{если } \mathbf{x}(n) \in \mathbf{C}_2. \end{cases}$$

4. Продолжение: $n:=n+1$ и к п.2

Многослойный Персептрон

Многослойный персептрон

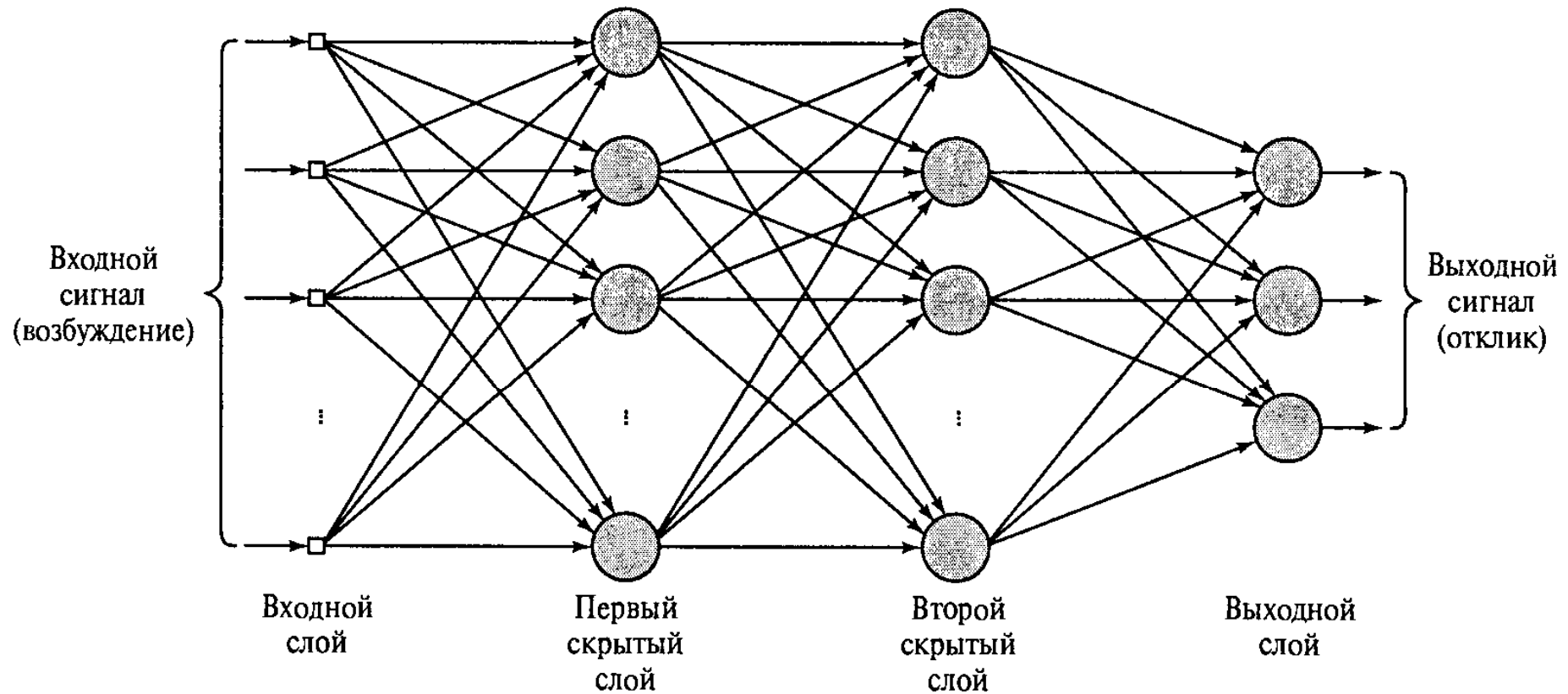


Рис. 4.1. Архитектурный граф многослойного персептрона с двумя скрытыми слоями

Потоки сигналов

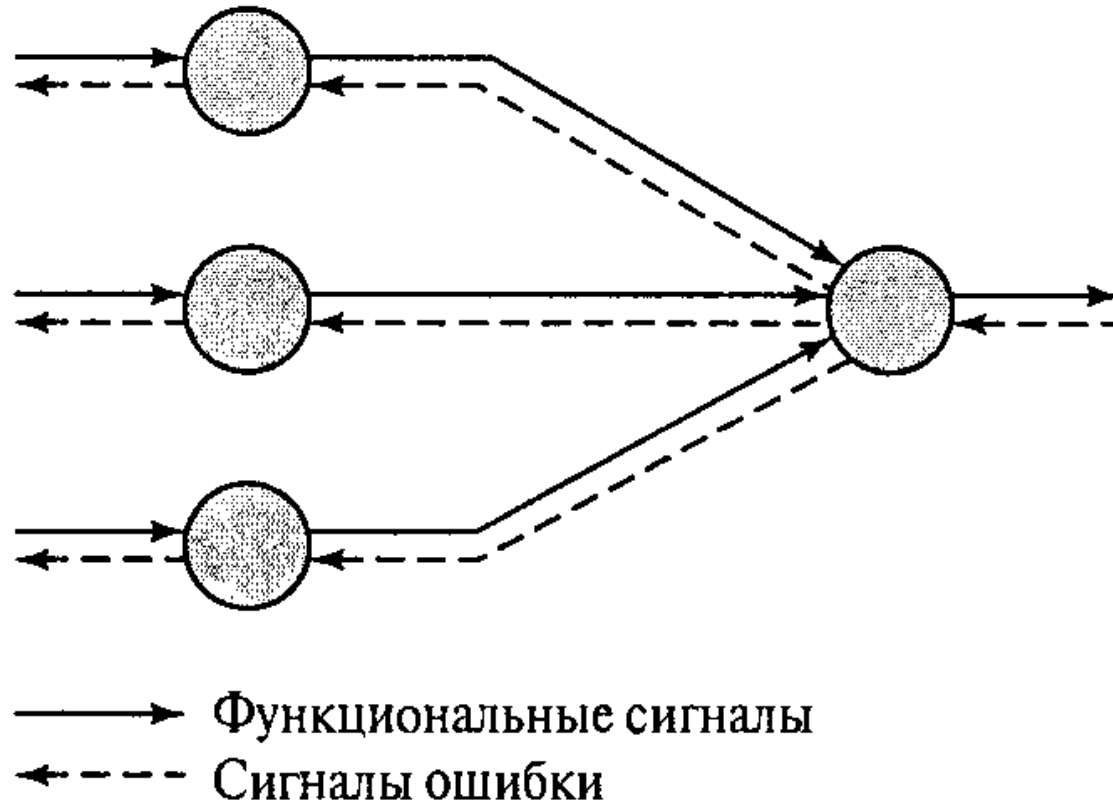


Рис. 4.2. Направление двух основных потоков сигнала для многослойного персептрона: прямое распространение функционального сигнала и обратное распространение сигнала ошибки

Алгоритм обратного распространения

- Предполагает два прохода
 - **Прямой:** входной вектор распространяется по сети от слоя к слою в прямом направлении, веса фиксированы
 - **Обратный:** все синаптические веса настраиваются по правилу коррекции ошибок

Математическая идея

- В сеть последовательно подаются обучающие наблюдения $(1, \dots, n, \dots, N)$
- На выходе сети получаем некоторый ответ $y_k(n)$ для k -го выходного нейрона

- Вычисляем ошибку полученного ответа от истинного

$$e_k(n) = d_k(n) - y_k(n)$$

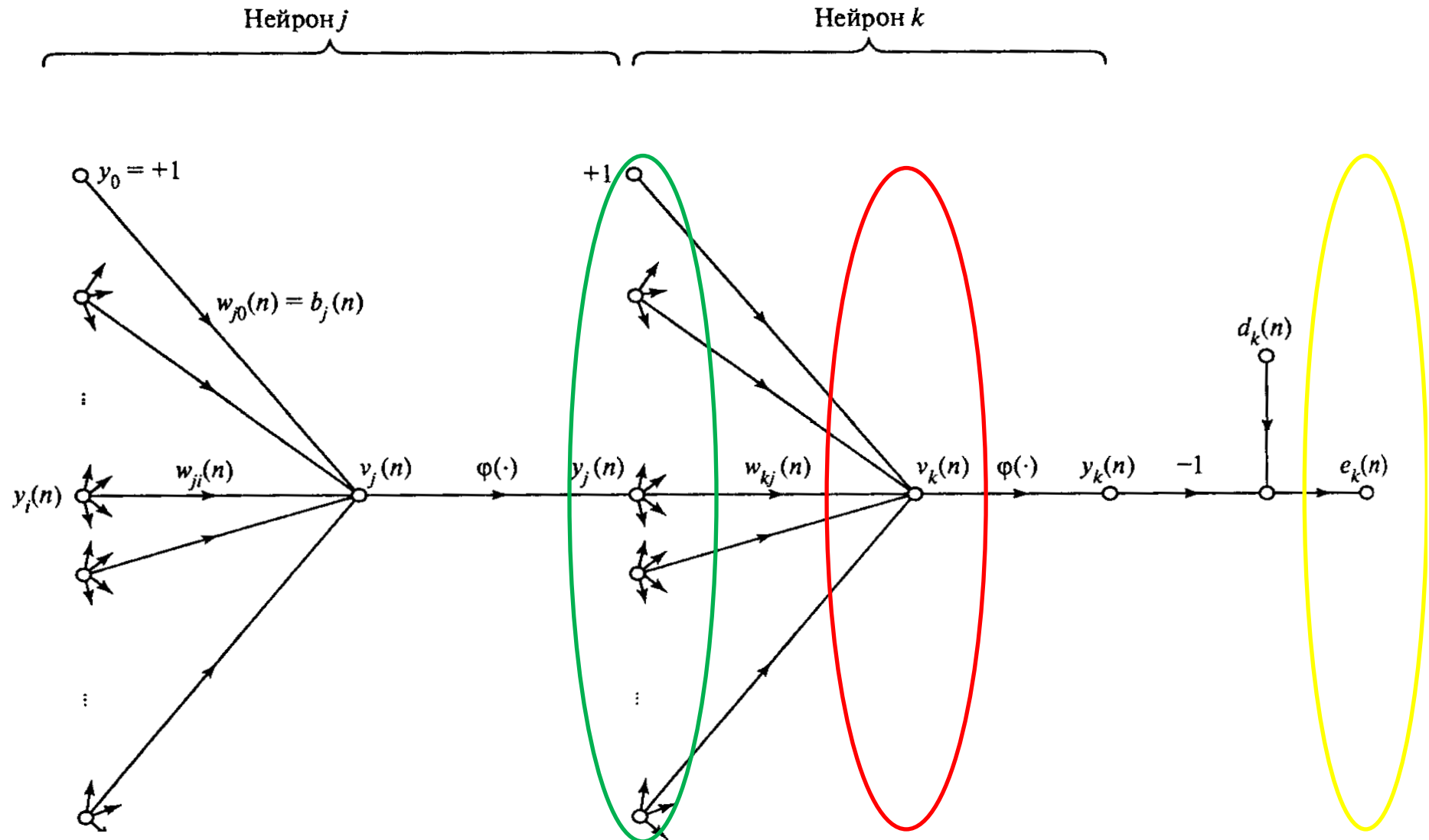
- Вычисляем общую энергию ошибки для всех выходных нейронов

$$E(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n)$$

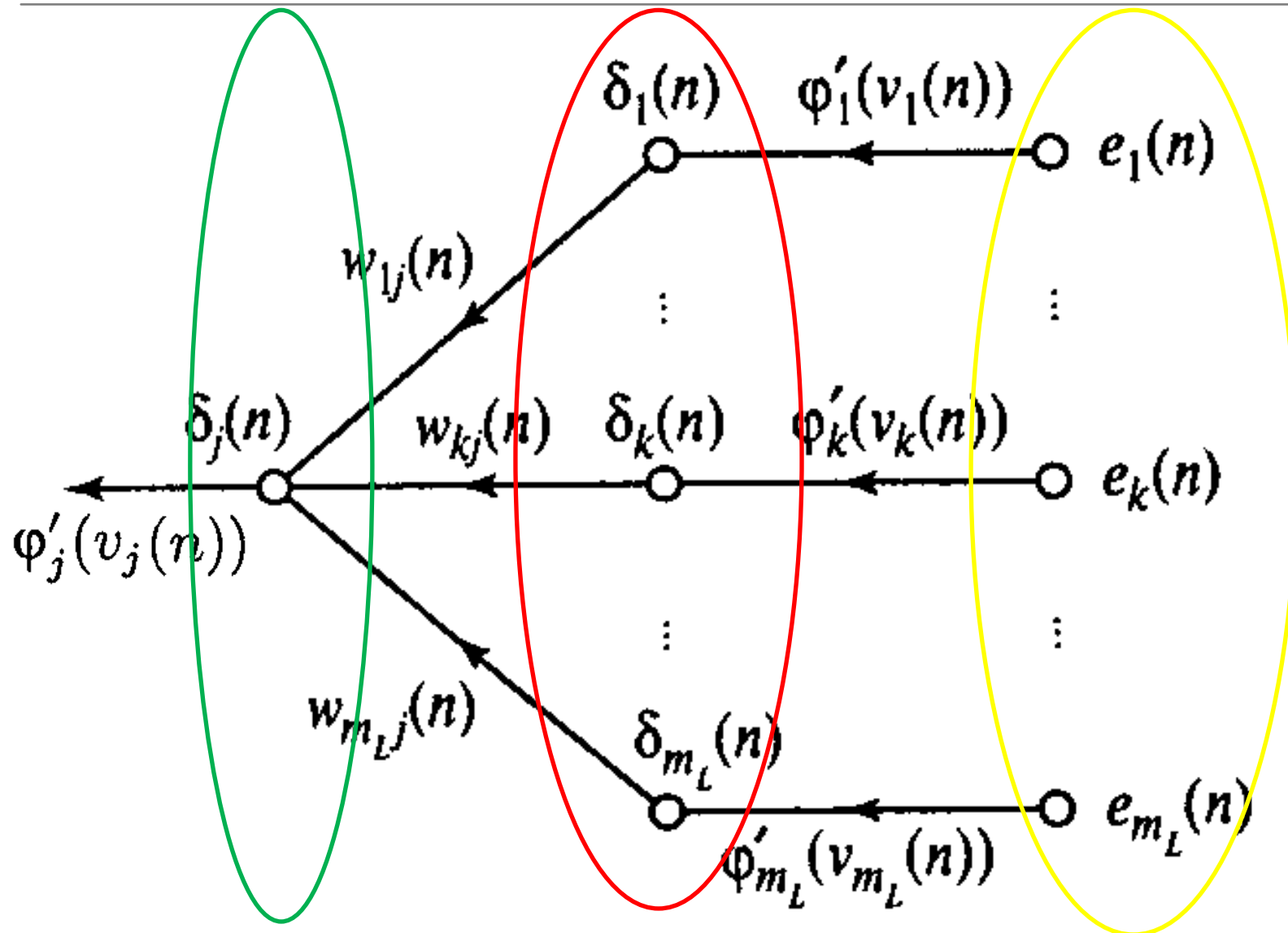
- Схастическим градиентным спуском минимизируем энергию ошибки, путем перенастройки весов

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}$$

Прямой проход



Обратный проход



Обратный проход

- Веса корректируются согласно **дельта-правилу**:

- Выходной слой:

$$\Delta w_{kj}(n) = \eta \delta_k(n) y_j(n)$$

- Скрытый слой:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

- **Локальный градиент δ** .

- Выходной слой:

$$\delta_k(n) = e_k(n) \varphi'_k(v_k(n))$$

- Скрытый слой:

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

Режимы обучения

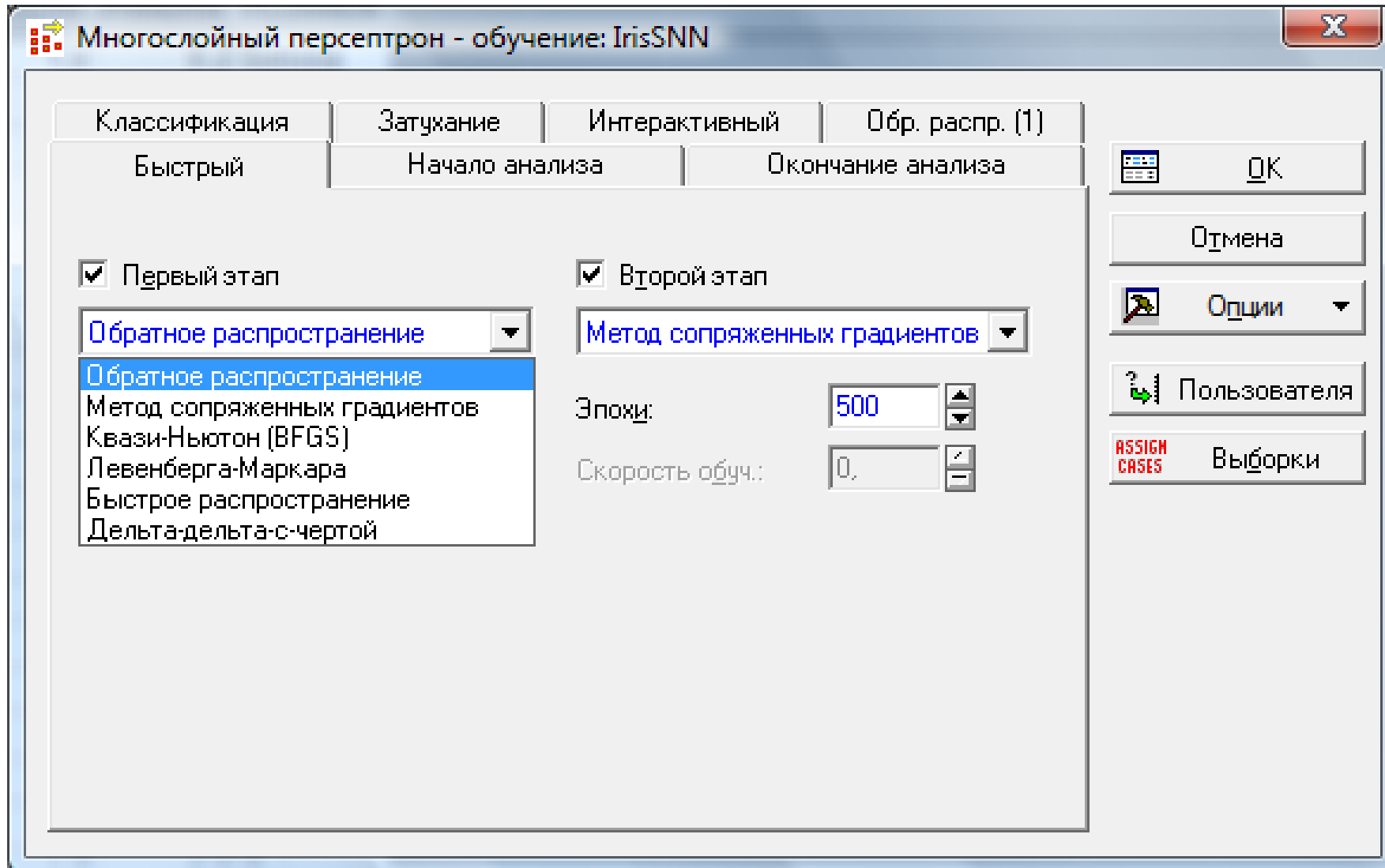
- **Последовательный** (для него все и выводилось)
 - Обучающие объекты пропускаем через сеть **по одному**
 - Веса корректируются для каждого объекта

- **Пакетный**

- Веса корректируются только после прохождения по сети **всего обучающего множества**

$$\Delta w_{ji}(n) = -\eta \frac{\partial E_{av}}{\partial w_{ji}} = -\frac{\eta}{N} \sum_{n=1}^N e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}$$

Алгоритмы и их реализация в STATISTICA



Многослойный перцептрон - обучение: IrisSNN

Классификация | Затухание | Интерактивный | Обр. распр. (1)

Быстрый | Начало анализа | Окончание анализа

Первый этап

Второй этап

Обратное распространение

- Обратное распространение
- Метод сопряженных градиентов
- Квази-Ньютон (BFGS)
- Левенберга-Маркара
- Быстрое распространение
- Дельта-дельта-с-чертой

Метод сопряженных градиентов

Эпохи: 500

Скорость обуч.: 0.

ОК

Отмена

Опции

Пользователя

ASSIGN CASES Выборки

Этапы обучения сети

- 1-й этап
 - Идея состоит в том, чтобы применить сначала некоторый алгоритм, не застревающий в локальных минимумах

- 2-й этап
 - После первого этапа лучше применять более точные алгоритмы поиска минимума

- Скорость обучения – скорость, с которой обучается алгоритмы ($\sim 0,1 - 0,9$)

- Эпоха – итерация алгоритма

Обратное распространение

- Режим: **Последовательный**
- Основан на **стохастическом градиентном спуске**
- Используется **обобщенное дельта-правило**:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

Константа **момента**

- Правило приводит к:
 - Ускорению спуска
 - Стабилизирующему эффекту

Метод Ньютона

- Основная идея – разложить функцию стоимости до 2-го порядка в точке $\mathbf{w}(n)$

$$\begin{aligned}\Delta \mathbf{E}(\mathbf{w}(n)) &= \mathbf{E}(\mathbf{w}(n+1)) - \mathbf{E}(\mathbf{w}(n)) = \\ &= \mathbf{g}^T(n) \Delta \mathbf{w}(n) + 1/2 \Delta \mathbf{w}^T(n) \mathbf{H}(n) \Delta \mathbf{w}(n)\end{aligned}$$

Матрица Гессе

- Из необходимого условия оптимальности:

$$\mathbf{g}(n) + \mathbf{H}(n) \Delta \mathbf{w}(n) = 0.$$

- Алгоритм Ньютона

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta \mathbf{w}(n) = \mathbf{w}(n) - \mathbf{H}^{-1}(n) \mathbf{g}(n)$$

Квази-Ньютон (BFGS)

- Режим: **Пакетный**

- В алгоритме Ньютона имеем:

$$\mathbf{w}(n + 1) = \mathbf{w}(n) + \Delta\mathbf{w}(n) = \mathbf{w}(n) - \mathbf{H}^{-1}(n)\mathbf{g}(n)$$

- **Но!** Т.к. матрица Гессе вычисляется трудно, строится итерационное приближение к обратной матрице Гессе по алгоритму ***BFGS (Бройдена-Флетчера-Гольдфарба-Шанно)***
- Это и есть итерационный алгоритм **Квази-Ньютона**

Метод Гаусса-Ньютона

- Функция стоимости:
$$\mathbf{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n e^2(i)$$

- Линеаризация $e(i)$

$$\mathbf{e}'(n, \mathbf{w}) = \mathbf{e}(n) + \mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n))$$

- Подставляя в функцию стоимости:

$$\begin{aligned} \frac{1}{2} \|\mathbf{e}'(n, \mathbf{w})\|^2 &= \frac{1}{2} \|\mathbf{e}(n)\|^2 + \mathbf{e}^T(n) \mathbf{J}(n) (\mathbf{w} - \mathbf{w}(n)) + \\ &+ \frac{1}{2} (\mathbf{w} - \mathbf{w}(n))^T \mathbf{J}^T(n) \mathbf{J}(n) (\mathbf{w} - \mathbf{w}(n)) \end{aligned}$$

Матрица Якоби

- Из необходимого условия получим алгоритм Гаусса-Ньютона

$$\mathbf{w}(n + 1) = \mathbf{w}(n) - (\mathbf{J}^T(n) \mathbf{J}(n))^{-1} \mathbf{J}^T(n) \mathbf{e}(n)$$

Левенберга-Маркара

- Режим: **Пакетный**
- Представляет собой модификацию метода Гаусса-Ньютона
- Чтобы можно было обращаться матрицу $\mathbf{J}^T(n)\mathbf{J}(n)$, к ней добавляется диагональная матрица

$$\mathbf{J}^T(n)\mathbf{J}(n) + \delta\mathbf{I}$$

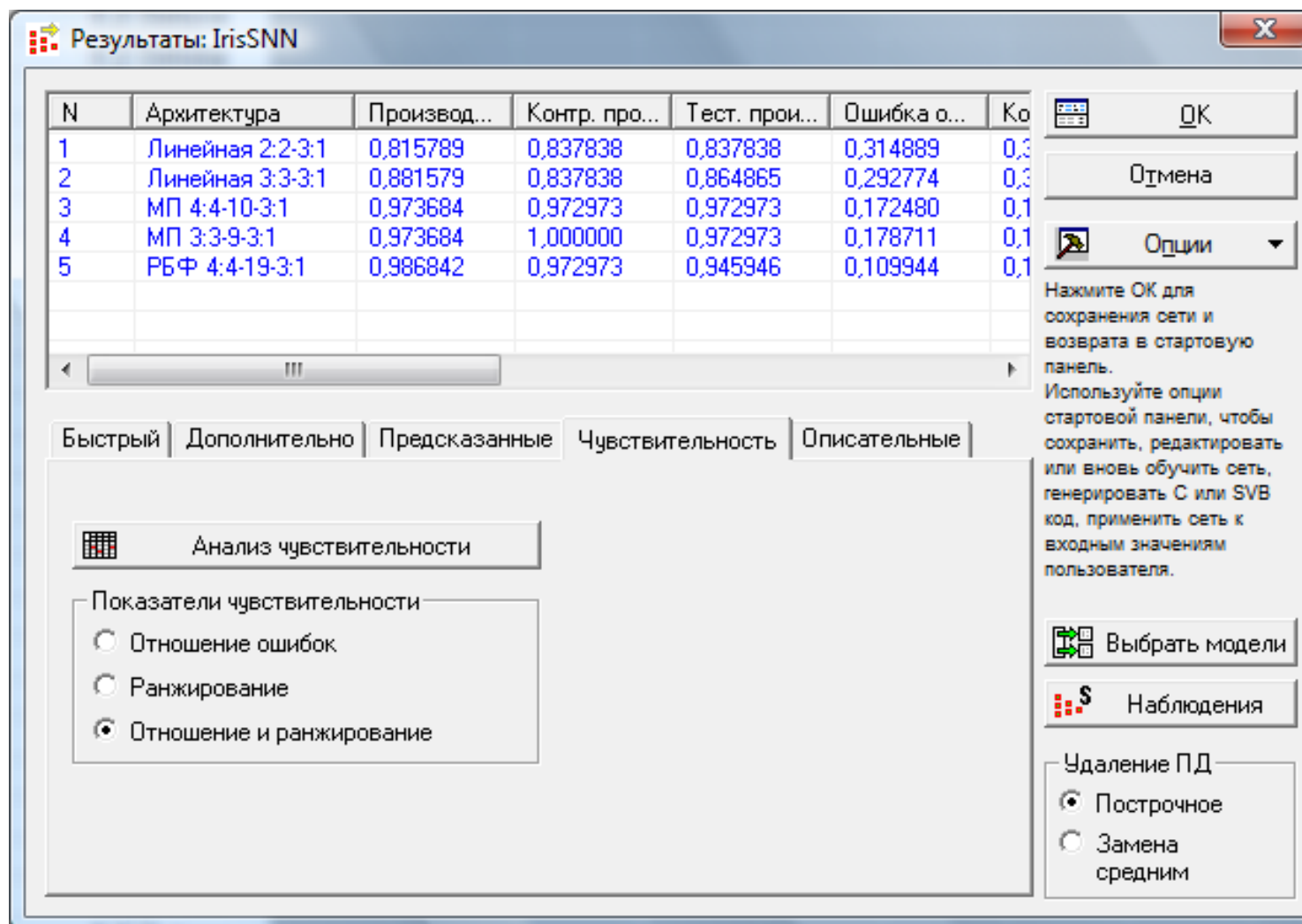
- Таким образом правило изменения весов выглялит так:

$$\mathbf{w}(n + 1) = \mathbf{w}(n) - (\mathbf{J}^T(n)\mathbf{J}(n) + \delta\mathbf{I})^{-1}\mathbf{J}^T(n)\mathbf{e}(n)$$

Дельта-Дельта-с-чертой

- Режим: **Пакетный**
- Это алгоритм обратного распространения но с той лишь разницей, что
 - Каждому синаптическому весу \rightarrow свой собственный параметр скорости обучения
 - Коэффициент инерции не используется

Анализ чувствительности доступен в следующем окне



N	Архитектура	Производ...	Контр. про...	Тест. прои...	Ошибка о...	Ко
1	Линейная 2:2-3:1	0,815789	0,837838	0,837838	0,314889	0,3
2	Линейная 3:3-3:1	0,881579	0,837838	0,864865	0,292774	0,3
3	МП 4:4-10-3:1	0,973684	0,972973	0,972973	0,172480	0,1
4	МП 3:3-9-3:1	0,973684	1,000000	0,972973	0,178711	0,1
5	РБФ 4:4-19-3:1	0,986842	0,972973	0,945946	0,109944	0,1

Быстрый | Дополнительно | Предсказанные | **Чувствительность** | Описательные

Анализ чувствительности

Показатели чувствительности

- Отношение ошибок
- Ранжирование
- Отношение и ранжирование

ОК

Отмена

Опции

Нажмите ОК для сохранения сети и возврата в стартовую панель. Используйте опции стартовой панели, чтобы сохранить, редактировать или вновь обучить сеть, генерировать S или SVB код, применить сеть к входным значениям пользователя.

Выбрать модели

Наблюдения

Удаление ПД

- Построчное
- Замена средним

Понятие чувствительности

- Чувствительность дает представление о том, **как входы влияют на выходы**

- Рассмотрим чувствительность обученной сети к переменной x_k :
 1. Сеть (**с переменной k**) прогоняем на тестовых данных –
> ошибка $E_{with\ k}$
 2. Сеть (**без переменной k**) прогоняем на тестовых данных
→ ошибка $E_{without\ k}$
 3. Чувствительность:

$$\delta_k = \frac{E_{without\ k}}{E_{with\ k}}$$

Переобучение сети

- Если сеть идеально работает на обучающей выборке, то это не значит что она хорошо обучилась
- Имеет место эффект переобучения
- Суть его состоит в следующем:

Сеть пытается так подогнать параметры, чтобы они идеально аппроксимировали функцию, но это приводит к сильной неустойчивости сети около обучающих объектов, и как следствие, не способности аппроксимировать функцию.

Переобучение сети

- Пример:
 - Можно записать в память информацию о том, к какому классу принадлежит наблюдение.
 - Тогда для любого обучающего наблюдения алгоритм моментально даст правильный ответ.
 - Но если мы подадим новое неизвестное наблюдение, то алгоритм вообще не даст ответа

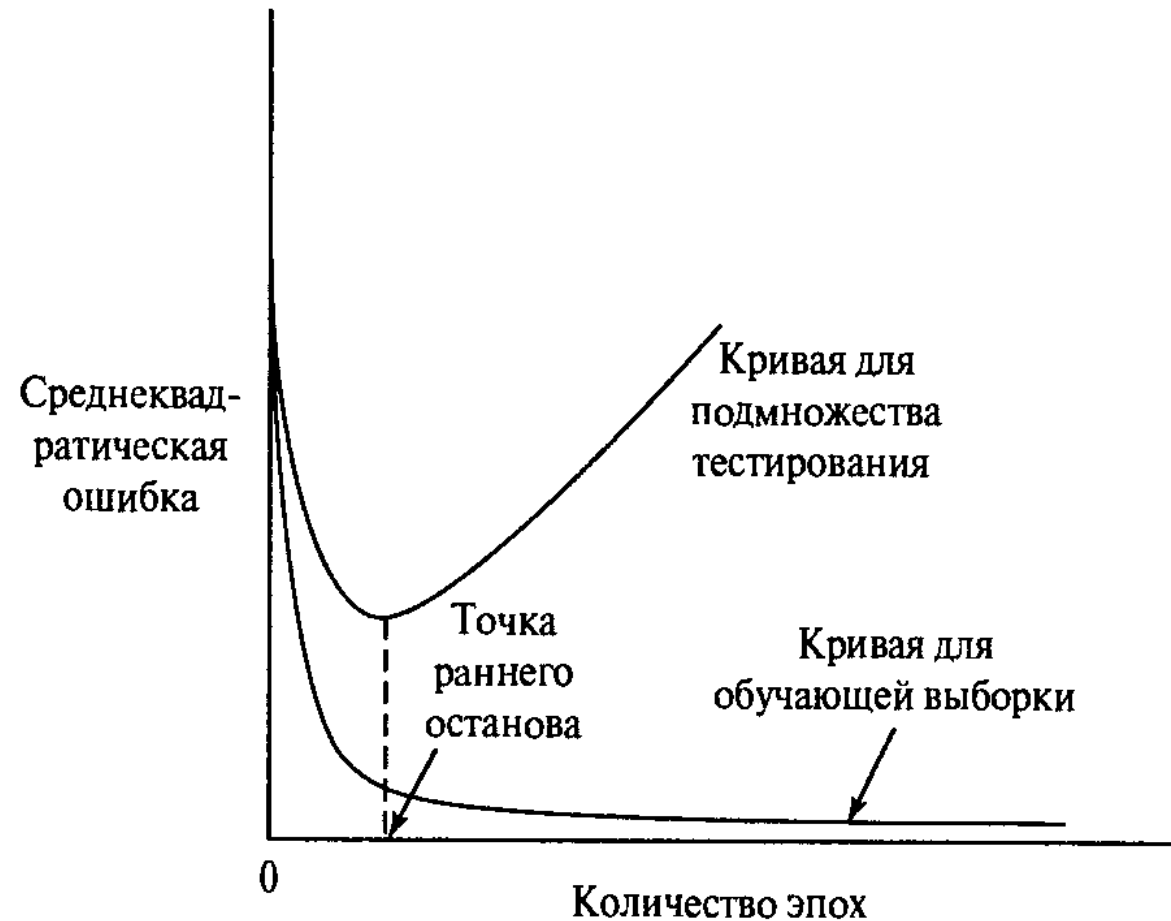
- Обучение «нулевое».

Пути преодоления переобучения (Все реализовано в SNN)

- Выбрать оптимальное число элементов сети (сколько??)
- Добавить шум к обучающим данным при обучении – сглаживает данные
- Ранняя остановка обучения
- Регуляризации сети (ограничение на структуру сети)

Обучение с ранним остановом (кросс-проверка)

- Обычно на практике наблюдается такая картина
- После точки раннего останова происходит переобучение сети
- Дальше точки раннего останова обучать сеть не стоит



Снижение сложности сети (регуляризация)

- Обычно, минимизируя ошибку сети $E_s(\mathbf{w})$, мы при этом не учитываем сложность сети
- Предлагается к этой ошибке добавить еще что-то, что накладывает ограничения на параметры сети

$$R(\mathbf{w}) = E_s(\mathbf{w}) + \lambda E_c(\mathbf{w})$$

Коэффициент регуляризации

Штраф за сложность

- Идея: сделать веса **меньше** → тем самым сделать подгоняющую функцию более гладкой

Регуляризация по Вигенду

- Регуляризация по Вигенду:

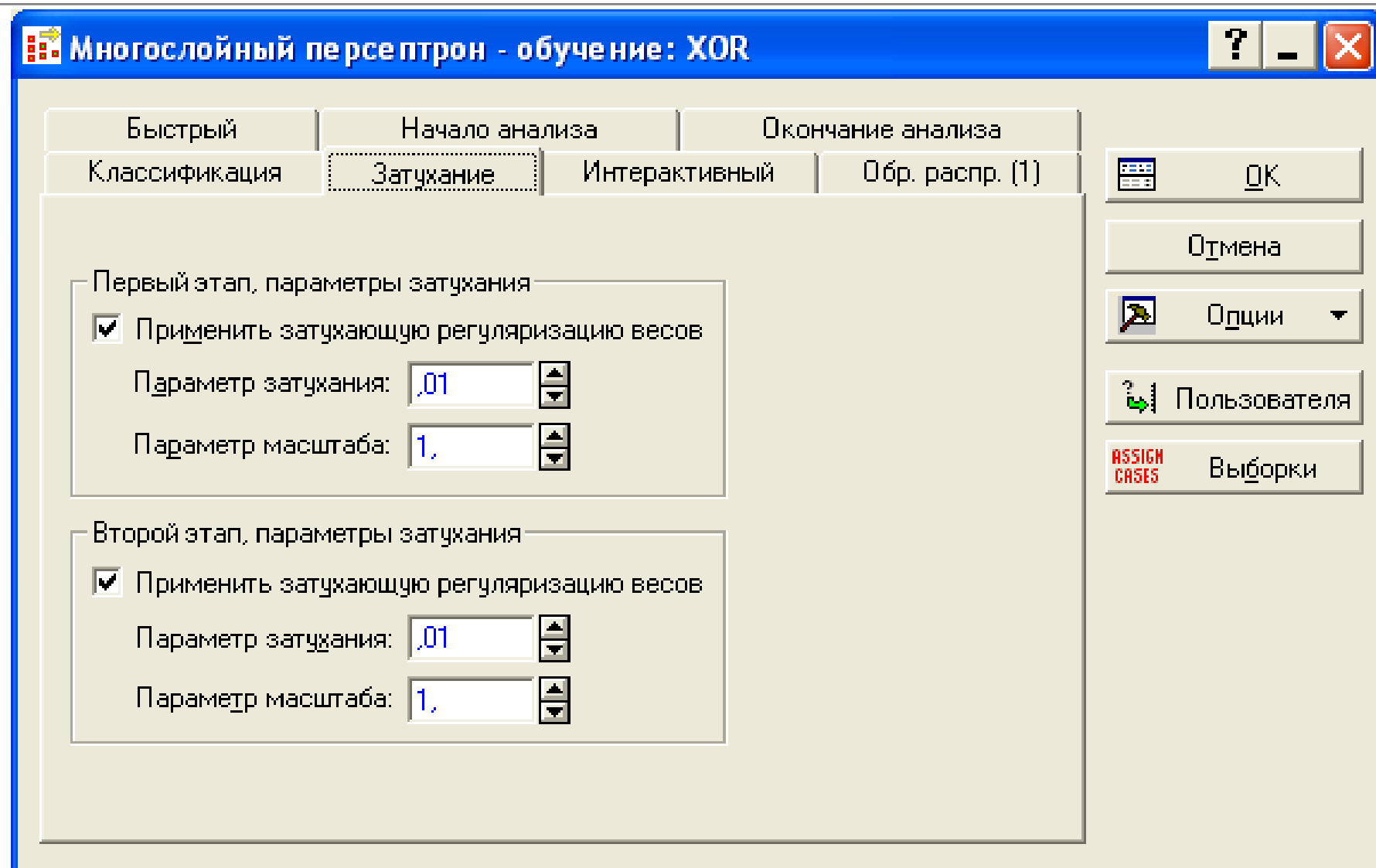
$$E_c(w) = \sum_{i \in C_{total}} \frac{w_i^2}{w_0^2 + w_i^2}$$

Множество всех
синаптических связей

Коэффициент масштаба

- Увеличиваются веса которые действительно важны для нейронной сети
- Связи с малыми весами можно удалить → тем самым упростив структуру сети

Регуляризация в SNN



Многослойный перцептрон - обучение: XOR

Быстрый | Начало анализа | Окончание анализа

Классификация | **Затухание** | Интерактивный | Обр. распр. (1)

Первый этап, параметры затухания

- Применить затухающую регуляризацию весов
- Параметр затухания:
- Параметр масштаба:

Второй этап, параметры затухания

- Применить затухающую регуляризацию весов
- Параметр затухания:
- Параметр масштаба:

ОК

Отмена

Опции

Пользователя

ASSIGN CASES | Выборки

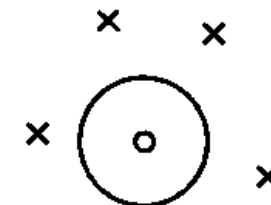
Сети на основе РБФ

Подход

- **Многослойный персептрон** пытается разбивать пространство наблюдений гиперплоскостями (прямые, плоскости,...).



- Иной подход – разбивать пространство гиперсферами (окружности, сферы,...). Гиперсфера задается своим центром и радиусом.

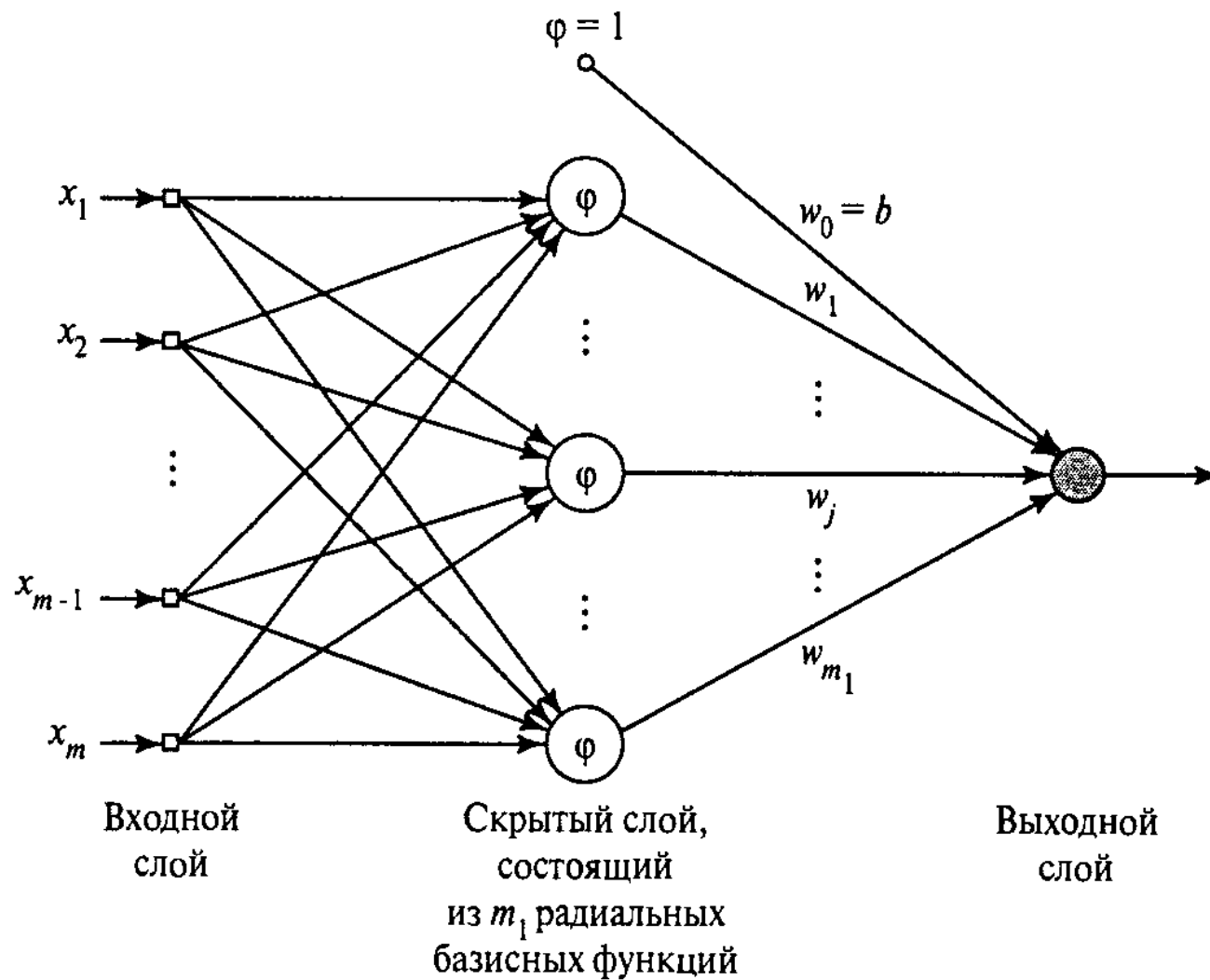


- Каждый объект относится к классу, которому принадлежит ближайший к нему центр.

Архитектура сети

- **Сеть состоит из 3-х слоев:**
 - **Входной:** из него подаются наблюдения, как обычно
 - **Скрытый:** каждый скрытый нейрон символизирует центр некоторой гиперсферы (кластера данных).
 - Входные веса – координаты центра.
 - Порог – радиус
 - Выход нейрона – реакция на расстояние от наблюдения до центра. (функция Гаусса)
 - **Выходной слой:** обычный однослойный персептрон

Архитектура сети



Функция Гаусса

- В скрытом слое, реакция на расстояние от центра нейрона до наблюдения задается функцией Гаусса:

$$G(\mathbf{x}, \mathbf{x}_i) = \exp \left(-\frac{1}{2\sigma_i^2} \|\mathbf{x} - \mathbf{x}_i\|^2 \right)$$

Где используются

- Сети RBF работают хорошо только с наблюдениями, «лежащими» близко к обучающим наблюдениям.
- При удалении от обучающего множества значение функции отклика быстро спадает до нуля.
- Используются в задачах
 - Регрессии
 - Классификации

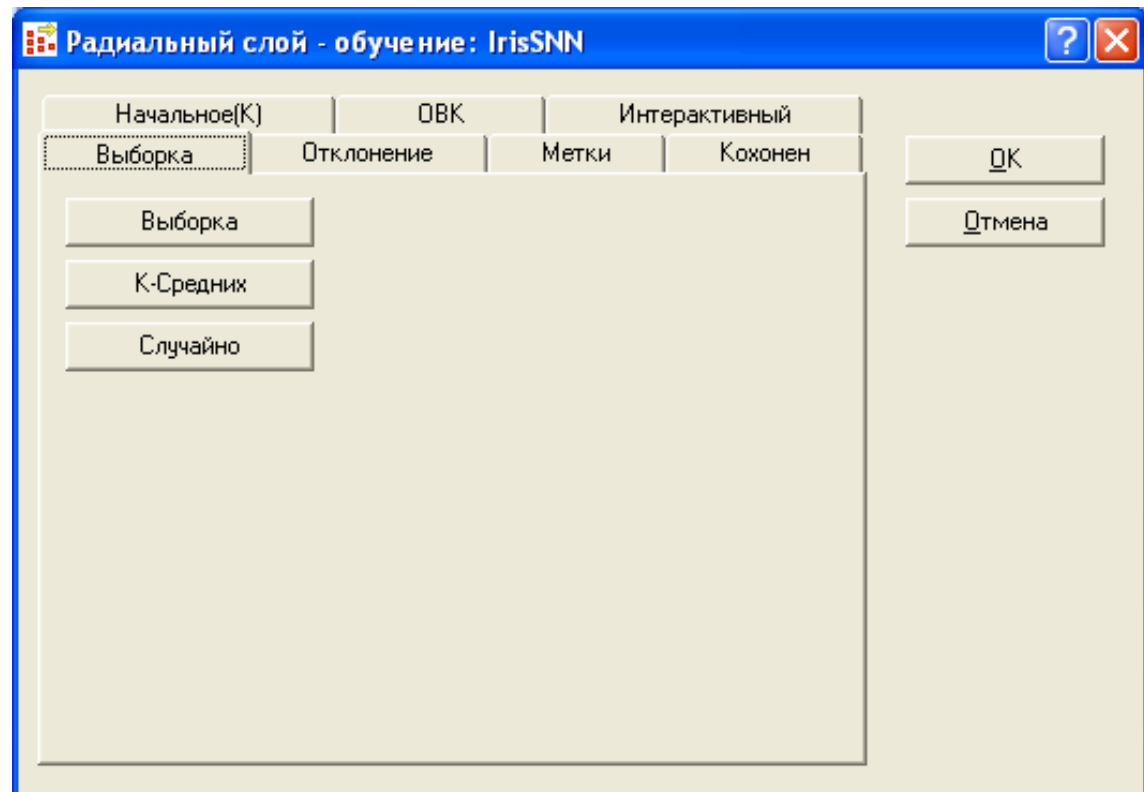
Обучение сети РБФ

- Обучение происходит в 2 этапа.
 - Сначала определяются центры и отклонения для радиальных элементов;
 - Затем настраиваются параметры линейного выходного слоя.

- Расположение центров должно соответствовать кластерам, реально присутствующим в исходных данных.

Определение центров

- В SNN реализованы 4 метода:
 - Выборка из выборки
 - Алгоритм К-средних
 - Случайно
 - Кохонен

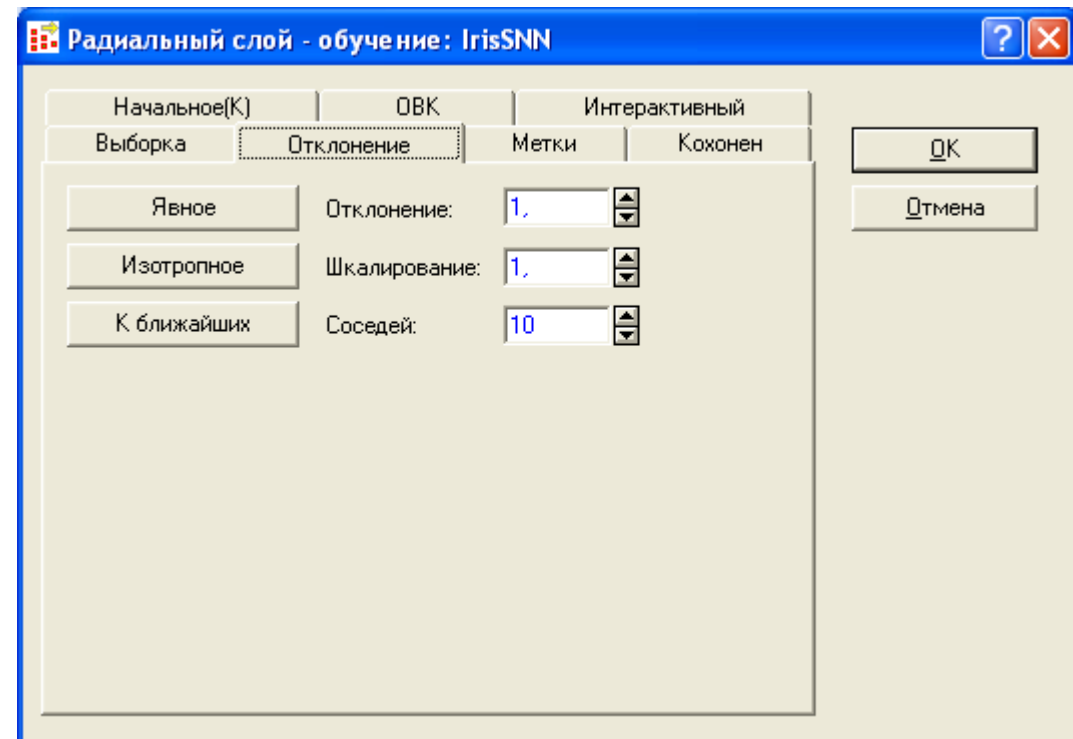


Алгоритм К-средних

1. Выбираются произвольно K -точек (K -центров)
2. Для каждого центра t_j определяется множество ближайших к нему обучающих наблюдений X_j
3. Центр t_j заменяется на центр масс точек X_j
4. И так повторять 2-3, пока не сойдется

Задание радиусов

- В SNN реализованы 3 метода:
 - Явное
 - Изотропное
 - К-ближайших соседей



Настройка параметров выходного слоя

- Когда выбраны центры и отклонения, параметры выходного слоя оптимизируются с помощью стандартного метода линейной оптимизации - алгоритма псевдообратных матриц (сингулярного разложения)
- Можно использовать пользовательские средства оптимизации (метод сопряженных градиентов). Они задаются в *Конструкторе сетей*

Обоснование

- Рассматривается задача интерполяции функции:

$$F(\mathbf{x}_i) = d_i, i = 1, 2, \dots, N.$$

- Метод радиальных базисных функций ищет функцию вида:

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} w_i \varphi_i(\mathbf{x})$$

$$\varphi_i(\mathbf{x}) = G(\|\mathbf{x} - \mathbf{t}_i\|), i = 1, 2, \dots, m_1$$

Обоснование

- Минимизируется энергия ошибки:

$$\mathbf{E}(F^*) = \sum_{i=1}^N \left(d_i - \sum_{j=1}^{m_1} w_j G(\|\mathbf{x}_i - \mathbf{t}_j\|) \right)^2$$

- Оптимальный вектор весов определяется в матричном виде:

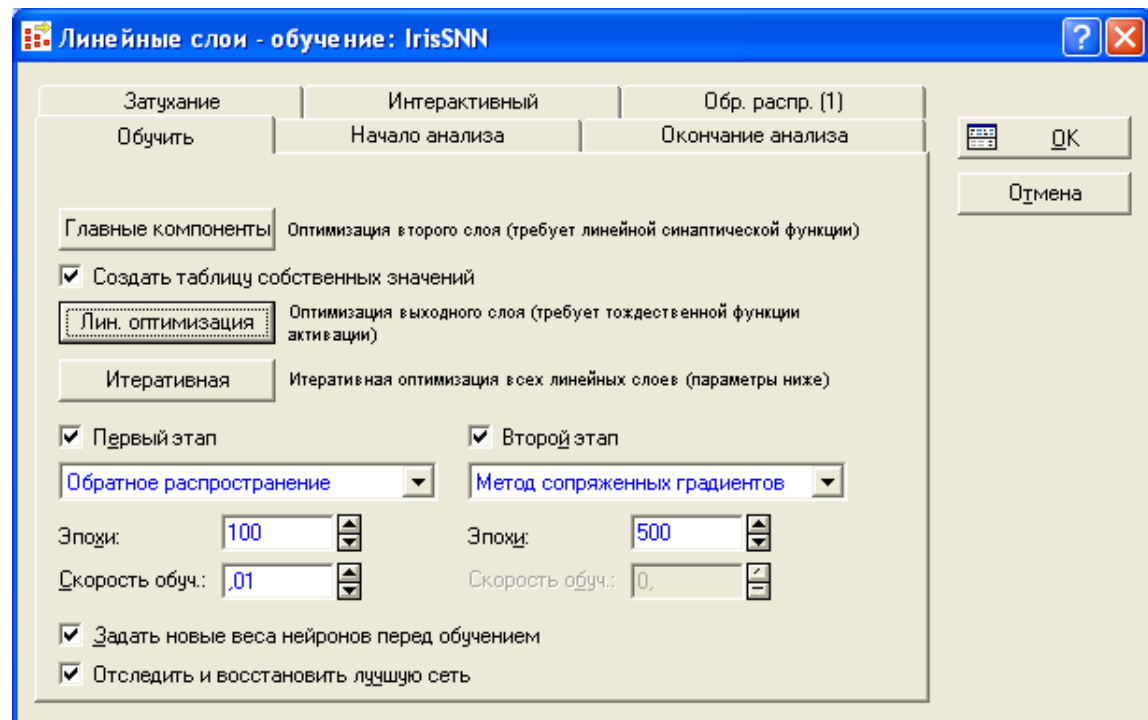
$$\mathbf{w} = \mathbf{G}^+ \mathbf{d}$$

где \mathbf{G}^+ – матрица, псевдообратная матрице \mathbf{G}

$$\mathbf{G}^+ = (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T$$

Метод псевдообратных (сингулярное разложение)

- Метод специально предназначен для вычисления псевдообратных матриц G^+ .
- Используется, когда выходной нейрон линейный (линейная функция активации)



Вероятностные Нейронные Сети (ВНС)

Идея

- Оценить плотность распределения наблюдений по обучающей выборке
- Подход основан на ядерных оценках
(Функция ядра – радиальная функция плотности, типа функции Гаусса)
- Если наблюдение расположено в данной точке, то в этой точке имеется некоторая плотность распределения

Отличие от РБФ

- Радиальные элементы берутся по одному на каждое обучающее наблюдение.
- Каждый из них представляет гауссову функцию с центром в этом наблюдении.
- Каждому классу соответствует один нейрон на выходном слое
- Каждый выходной нейрон соединен со всеми радиальными элементами, относящимися к его классу (ост. связи = 0)

Отличие от сети РБФ

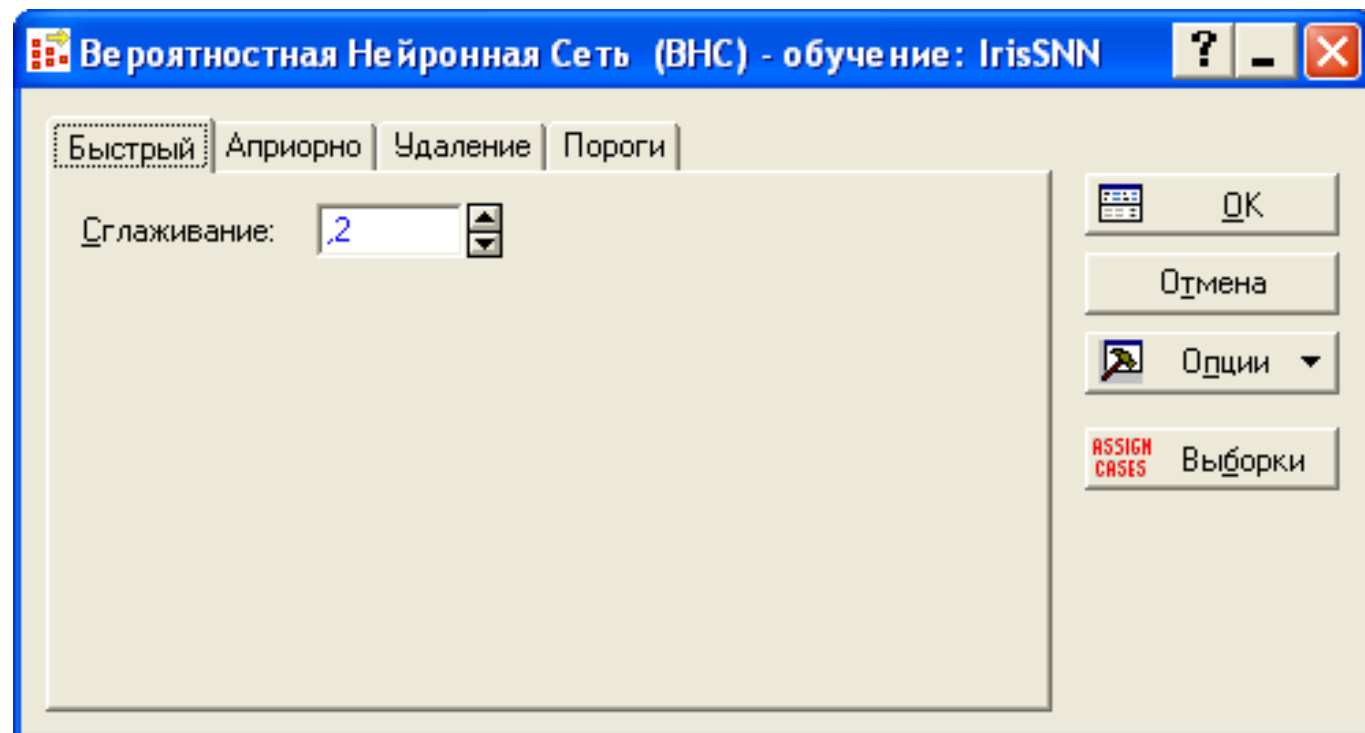
- Выходной сигнал ~ ядерным оценкам вероятности принадлежности соответствующим классам
- Этот сигнал нормируется функцией активации

$$\frac{x}{\sum_i x_i}$$

- Получаются окончательные оценки вероятности принадлежности классам.
- Далее, можно добавить еще 4-й слой – чтобы учесть матрицу потерь. На выходе выбирается тот класс у которого минимальная цена ошибки.

Параметры

- Степень сглаживания – это степень размытости «купола» (или просто радиус кластера) (обычно $\sim 0.1 - 3$)
- Задание пропорций классов (Априорные вероятности)



Где используется

- **Задача классификации**
- На выходе дает вероятность принадлежности к каждому классу
- Очень быстро обучается и работает
- Но требуют много памяти (на радиальном слое центры нейронов – все входные наблюдения)
- Используются в генетических алгоритмах отбора переменных
- Не способны экстраполировать

Обобщенно-Регрессионные Нейронные Сети (ОРНС)

Идея

- ОРНС аналогична ВНС, но предназначена для задачи **регрессии.**

Модель нелинейной регрессии

- Рассматривается модель:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, i = 1, 2, \dots, N.$$

- Требуется оценить функцию $f(\mathbf{x})$, которая задается в виде:

$$f(\mathbf{x}) = E[y|\mathbf{x}]$$

Оценка функции регрессии

- Оценка функции регрессии *Надара-Ватсона*:

$$F(\mathbf{x}) = \sum_{i=1}^N W_{N,i}(\mathbf{x}) y_i$$

где нормированная функция весов:

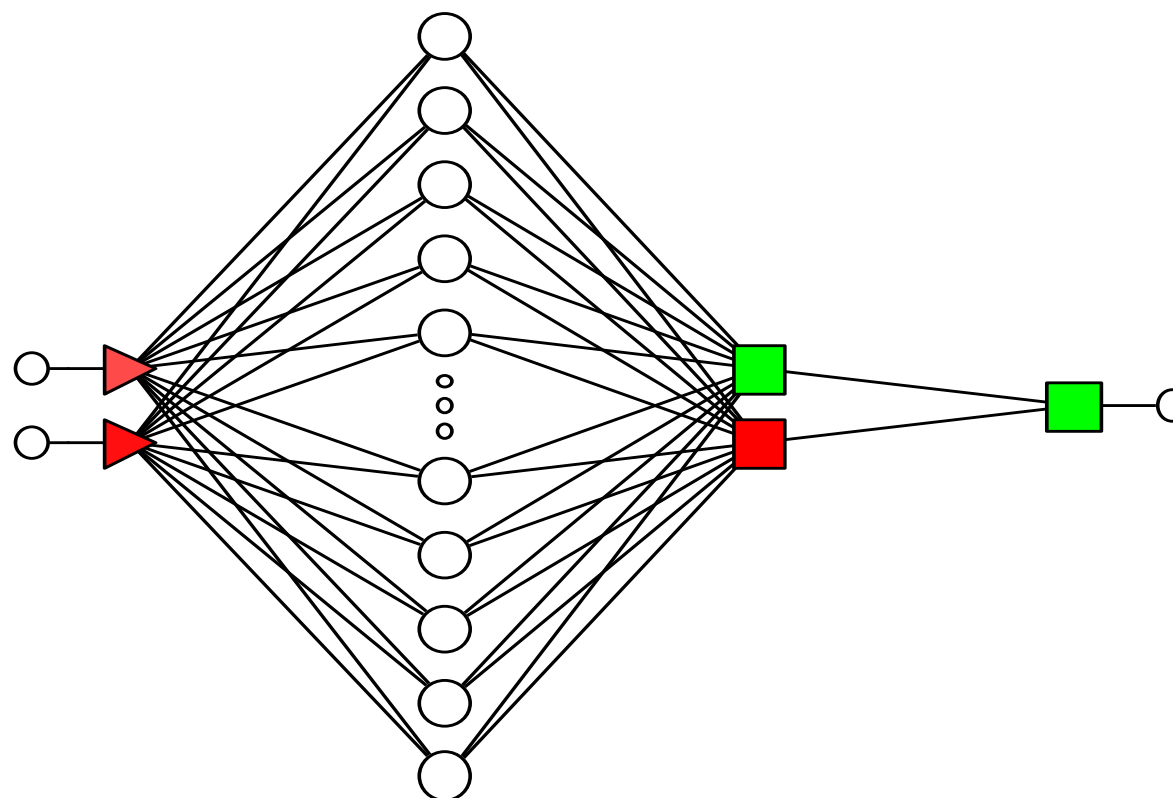
$$W_{N,i}(\mathbf{x}) = \frac{K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}{\sum_{j=1}^N K\left(\frac{\mathbf{x}-\mathbf{x}_j}{h}\right)}, \quad i = 1, 2, \dots, N$$

- Каждому значению y_i соответствует некоторый вес, чем ближе \mathbf{x}_i к \mathbf{x} , тем этот вес больше

Архитектура ОРНС

- 1-й слой: как обычно, это входные элементы
- Радиальный (или 2-й) слой: состоит из всех обучающих наблюдений. На выходе 2-го слоя – значения функции ядра
- 3-й слой:
 - 1-й нейрон вычисляет взвешенную сумму S_w
 - 2-й нейрон вычисляет нормировочный коэффициент N (сумма ядерных функций)
 - (Вообще нейронов может быть больше, но нормировочный один!)
- 4-й слой: просто вычисляет отношение 2-х входных значений (S_w / N)

Архитектура



Где используется

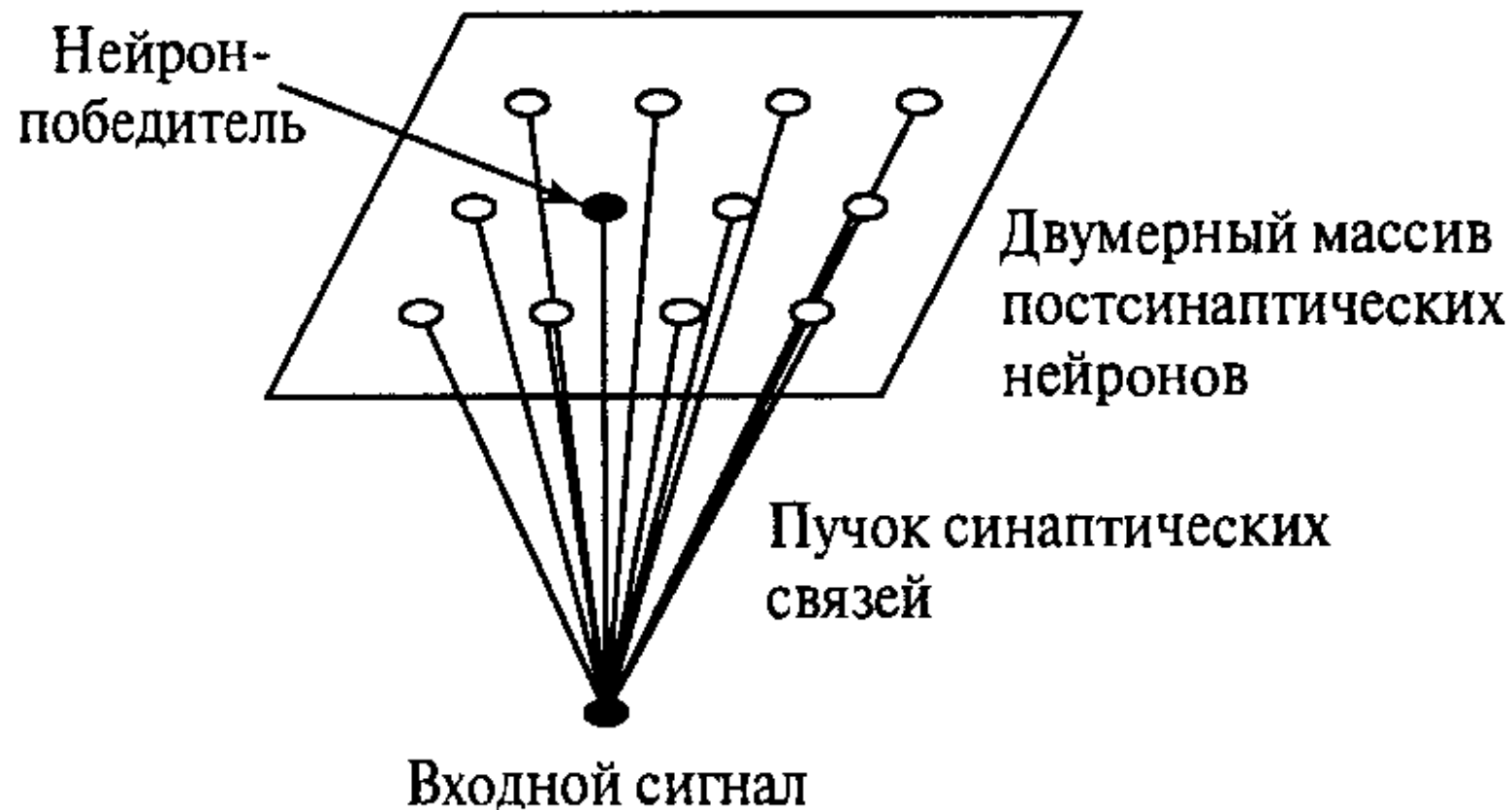
- Задачи регрессии
- Во всем остальном аналогичны ВНС

Карты самоорганизации

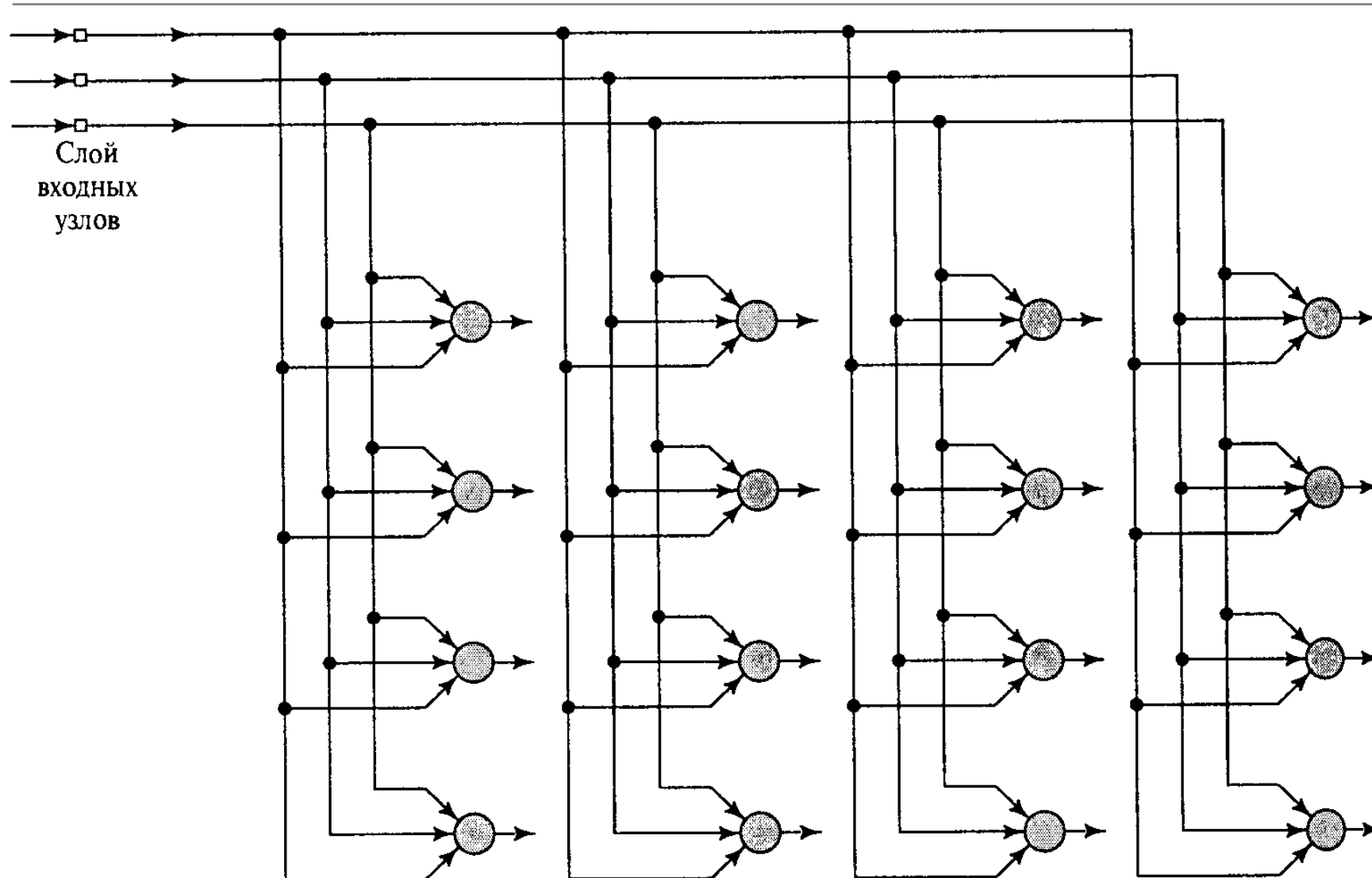
Идея

- Имеются многомерные данные
- Выбирается несколько произвольных точек из этого пространства (как в сетях РБФ)
- Каждая выбранная точка интерпретирует некоторый нейрон.
- Эти нейроны условно располагают на плоскости (топологическая карта), для наглядности.
- Затем итеративно в многомерном случае центры нейронов (выбранные точки) смещаются в области где данных больше, параллельно нейроны смещаются и на топологической карте.
- В результате, элементы, соответствующие центрам, расположенным близко друг от друга в пространстве входов, будут располагаться близко друг от друга и на топологической карте.

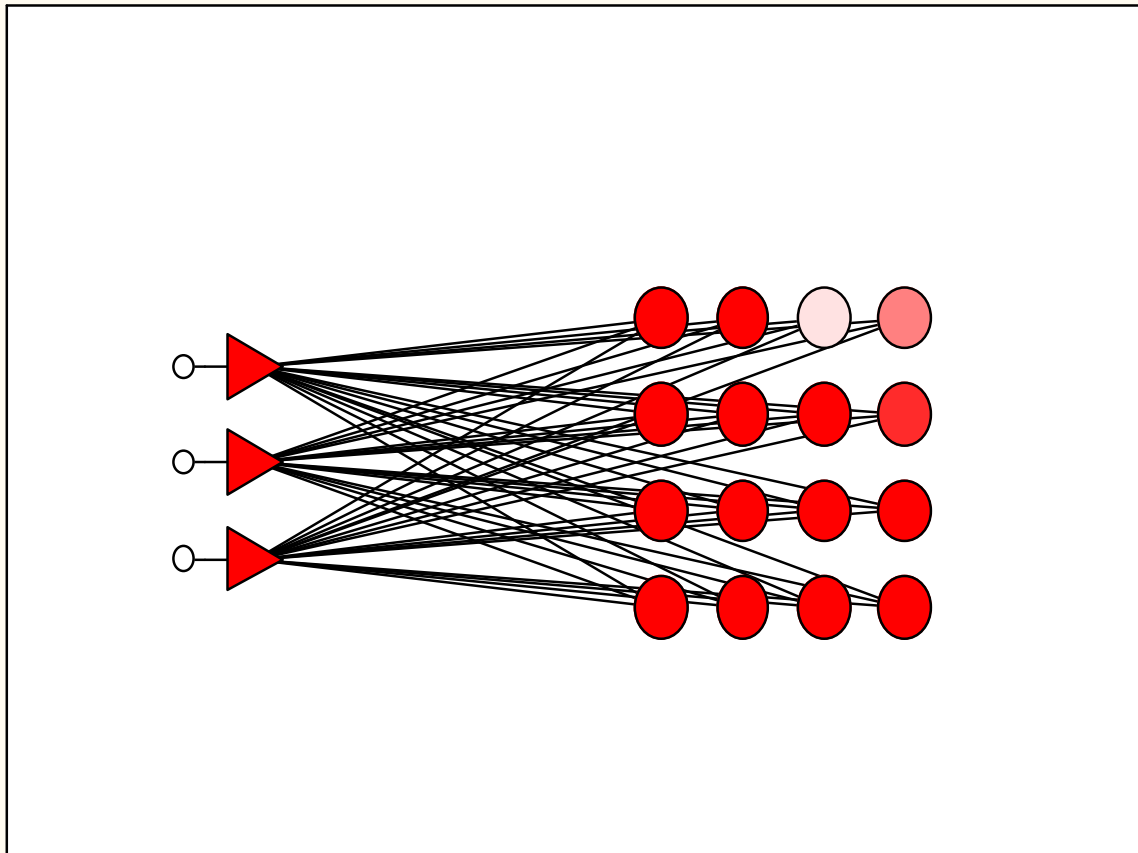
Модель Кохонена



Представление сетью



Архитектура в SNN



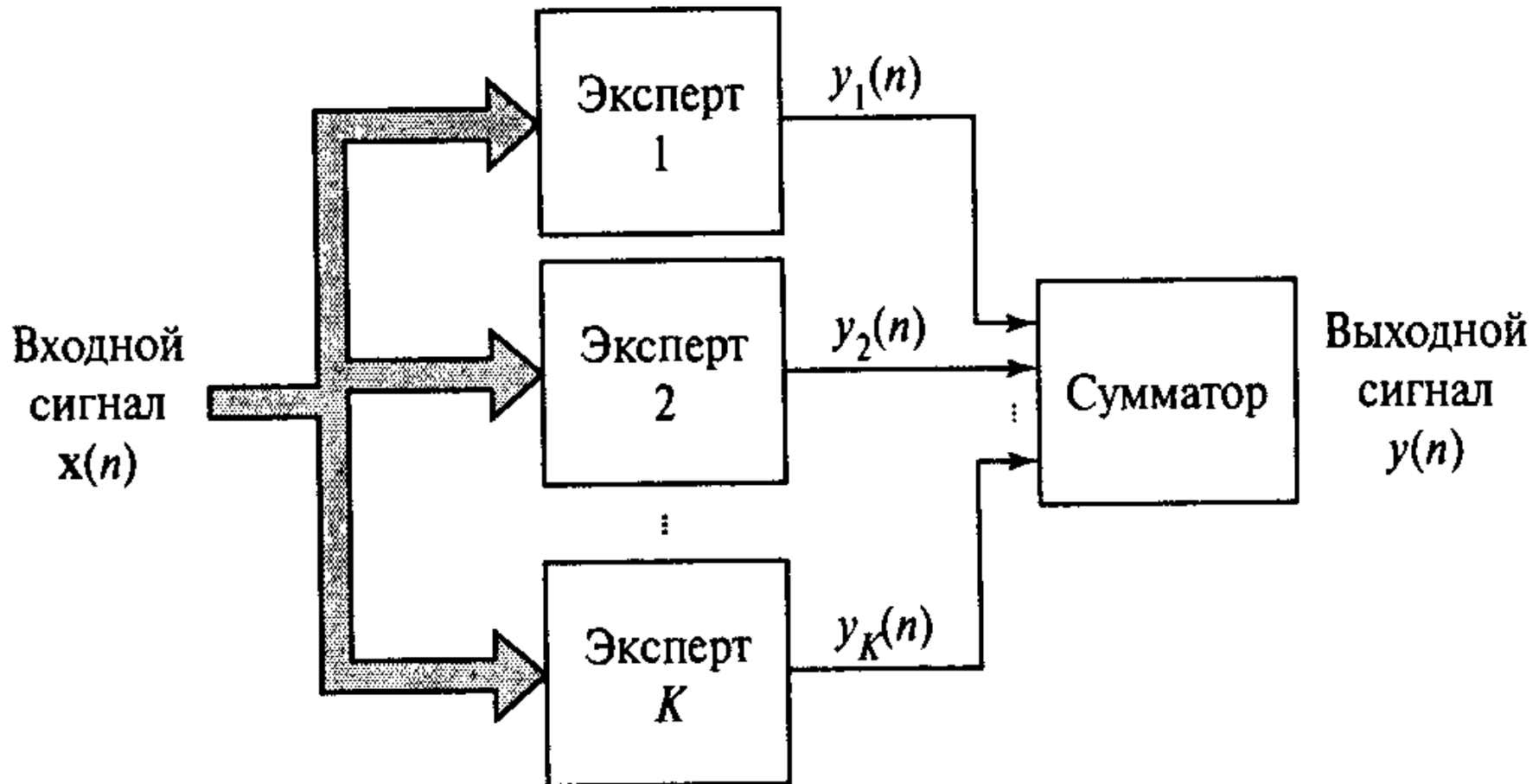
- 2 слоя:
 - Входной
 - Выходной (топологическая карта)

Где используются

- Сети Кохонена используются в задачах **кластеризации**.

Ансамбли Сетей

Общий вид ансамбля



- Эксперт – отдельная нейронная сеть с выходным сигналом $y(n)$.

Выходные ансамбли

- Можно сочетать любые наборы сетей
- Если сети имеют различные выходы, то ансамбль имеет несколько выходов
- Если выход общий, то
 - В задаче классификации: выходной принцип «победитель берет все»
 - В задаче регрессии: усреднение выходов по сетям

Доверительные ансамбли

- Используются только в задачах классификации
- Оценивают общие доверительные уровни для всех классов, а не просто выбирают лучший класс

Спасибо за внимание!

- Закажите курсы по нейронным сетям в Академии
Анализа Данных СтатСофт

http://www.statsoft.ru/statportal/tabID_38/DesktopDefault.aspx#all

- Уникальная книга Нейронные Сети STATISTICA SNN,
2-е издание, переработанное и дополненное

http://www.statsoft.ru/statportal/tabID_45/DesktopDefault.aspx

- Исчерпывающая информация по курсам СтатСофт:
тел. 8 495 7877733, факс 8 495 9160393

www.statsoft.ru

- Пишите: sale@statsoft.ru