

Московский авиационный институт (МАИ)

Факультет прикладной математики и физики



**Кафедра вычислительной математики
и программирования**



Нейроинформатика

Ю.В.Тюменцев

Предмет нейроинформатики и цели курса (I)

Предмет нейроинформатики — нейросетевые модели (НС-модели), их виды, способы создания и использования.

НС-модели — математические модели, основанные на использовании *искусственных нейронных сетей*.

Нейроинформатика — «две стороны одной медали»:

- как часть вычислительной математики (**формирование НС-моделей**);
- как часть информатики (компьютерная **реализация** НС-моделей).

Двойственность моделей и систем в их современном виде — они могут быть совершенно *одинаковыми* на уровнях **синтаксическом** (как модель/система устроена) и **семантическом** (какое поведение реализует модель/система), а *различаться* только на уровне **прагматическом** (с какой целью модель/система реализует свое поведение).

Предмет нейроинформатики и цели курса (II)

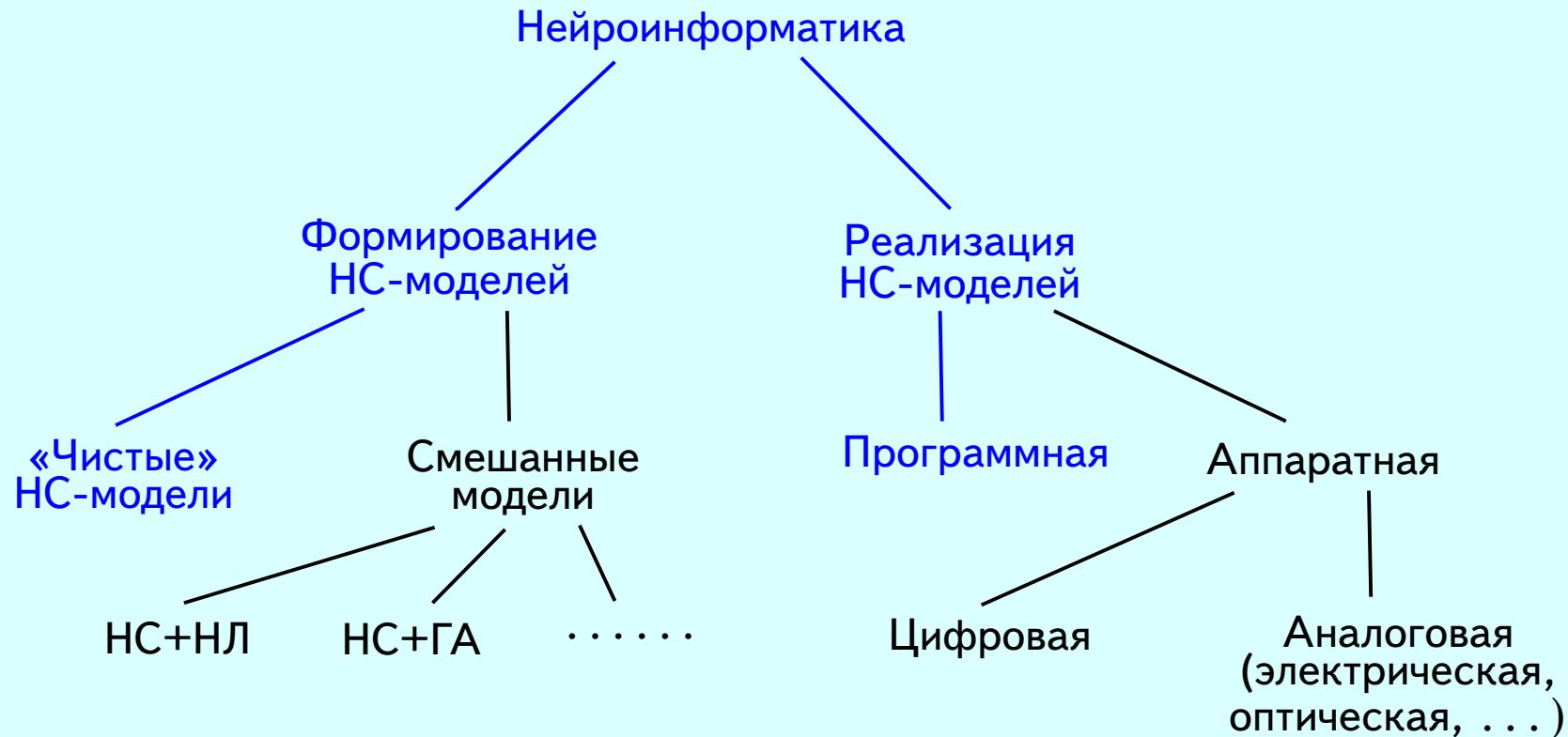
Цели курса:

1. Дать **представление** об основных классах НС-моделей, способах их формирования и использования.
2. Получить практические **навыки** работы с важнейшими видами НС-моделей.

Инструментарий:

1. **НС-модели** — аппарат математического моделирования.
2. **Практика** — система MATLAB с пакетом NEURAL NETWORK.

Предмет нейроинформатики и цели курса (III)



Смешанные модели: НС+НЛ, НС+ГА, НС+СОЗ, НС+МА, НС+ММ, НС+НЛ+ГА, НС+МА+ГА, НС+МА+СОЗ, НС+НЛ+СОЗ+МА, НС+НЛ+ГА+МА+СОЗ+ММ и т.п.

НС — нейросетевые модели, **НЛ** — модели нечетких систем, **ГА** — модели эволюционных вычислений, **СОЗ** — модели систем, основанных на знаниях, **МА** — мультиагентные модели, **ММ** — традиционные вычислительные модели.

НС-моделирование — зачем оно?

Чего нам **не хватает** в других («традиционных») подходах к математическому моделированию?

Основные претензии к традиционным моделям:

- 1) недостаточная **гибкость** моделей, обусловленная недостаточным числом «степеней свободы» в них;
- 2) необходимость **исчерпывающе полной** реализации моделей (требуется детально описать не только **что** надо сделать, но и **как** это надо делать);
- 3) отсутствие **адаптивности** (возможности приспосабливаться к ситуации, изменившейся *непредсказуемым* образом);
- 4) плохая приспособленность моделей к работе в условиях **неопределенности** (объект + среда), особенно невероятностных типов;
- 5) низкий уровень **отказоустойчивости** (fault-tolerance) и устойчивости к повреждениям (damage-tolerance) моделей, несмотря на использование дорогостоящего резервирования их элементов.

Что дает НС-моделирование?

НС-моделирование позволяет обеспечить:

- 1) очень высокую **гибкость**, практически неограниченное число «степеней свободы»;
- 2) замену детальной реализации (через программирование) модели на ее **обучение** (исключается необходимость детального описания того, **как** надо решать задачу);
- 3) очень высокую, практически неограниченную, степень **адаптивности**;
- 4) высокую приспособленность к работе в условиях **неопределенности**, особенно при совместном использовании НС-моделирования с методами из других областей вычислительного и искусственного интеллекта (computational intelligence & artificial intelligence);
- 5) очень высокий уровень **устойчивости к отказам и повреждениям**, обеспечиваемый *не резервированием* (т.е. введением элементов, не требуемых непосредственно для решения соответствующей прикладной задачи), *а структурной организацией* НС-модели.

Прагматика задач НС-моделирования (I)

Два вида задач НС-моделирования:

- ❑ моделирование процессов в **биологических** нейронных сетях (ЕНС);
- ❑ моделирование процессов в **искусственных** нейронных сетях (ИНС).

Целевая аудитория видов НС-моделирования:

- ❑ **ЕНС-модели** — нейробиология;
- ❑ **ИНС-модели** — математика, физика, химия, инженерные науки, экономика и т.п.

Прагматика задач НС-моделирования (II)

У ЕНС-моделирования и ИНС-моделирования различные **цели**:

- ЕНС-модели** — понять, как работают биологические нейронные сети, нервная система, мозг;
- ИНС-модели** — получить **средство решения** прикладных задач в различных областях (физика, инженерные науки, экономика и т.п.).

К этим видам моделей предъявляются также и различные **требования**:

- ЕНС-модели** — максимально возможное «**биологическое правдоподобие**»;
- ИНС-модели** — практическая применимость и **эффективность** в различных прикладных областях (биологическое правдоподобие не требуется).

Предмет курса — **искусственные нейронные сети** и процессы в них.

Требования к НС-моделированию

НС-моделирование должно обеспечивать:

1. Очень высокую **гибкость**, практически неограниченное число «степеней свободы» в формируемых моделях.
2. Замену детальной реализации (через программирование) модели на ее формирование путем **обучения**.
3. Очень высокую, практически неограниченную, степень **адаптивности**.
4. Высокую приспособленность к работе в условиях **неопределенности**.
5. Очень высокий уровень **устойчивости к отказам и повреждениям**.

Краеугольные камни НС-моделирования

НС-моделирование основывается на следующих «генеральных идеях»:

- 1.** Представление зависимостей от многих величин через зависимости от меньшего числа величин, в пределе — **от одной величины**.
- 2.** Сетевая (**композиционная**) связь между величинами в модели.
- 3.** Использование **обучения** как инструмента формирования и настройки модели.

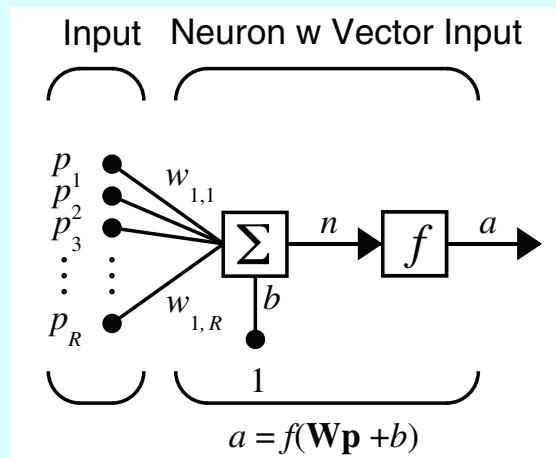
Наиболее общая трактовка **НС-моделей** — композиция *отображений-примитивов*, реализующая предписанную *системную динамику* (различается для этапов обучения и рабочего использования).

Коннекционизм — свойства НС-модели почти целиком определяются *структурой и Весами ее связей*. Конкретный вид связываемых элементов (нейронов) не имеет принципиального значения, требуется только наличие какой-либо (любой) *нелинейности*.

Обучение (варианты подходов): обучение с учителем, обучение без учителя (самообучение), обучение с подкреплением. **Цель обучения** — минимизация некоторой *функции ошибки*. Вид функции ошибки существенно влияет на свойства сети.

Основные структурные элементы НС-моделей (I)

Нейрон – 1



Алгоритмы, реализуемые нейроном

$$n = b + w_1 p_1 + \dots + w_n p_n = b + \sum_{i=1}^n w_i p_i,$$

$$a = f(n) = f(b + \sum_{i=1}^n w_i p_i),$$

$$b = w_0 p_0 = b \cdot 1, \quad n = \sum_{i=0}^n w_i p_i.$$

p_i — входы нейрона, a — **единственный** выход нейрона, $w_{1,i}$ — синаптические веса связей нейрона, b — смещение, $\Sigma(\cdot)$ — аддитивный сумматор, $f(\cdot)$ — активационная функция

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Fig.2-5, p.2-7).

Основные структурные элементы НС-моделей (II)

Активационные функции – 1:

- линейная функция

$$f(V) = cV ;$$

- логистическая функция

$$f(V) = \frac{1}{1 + e^{-cV}} ;$$

- гиперболический тангенс

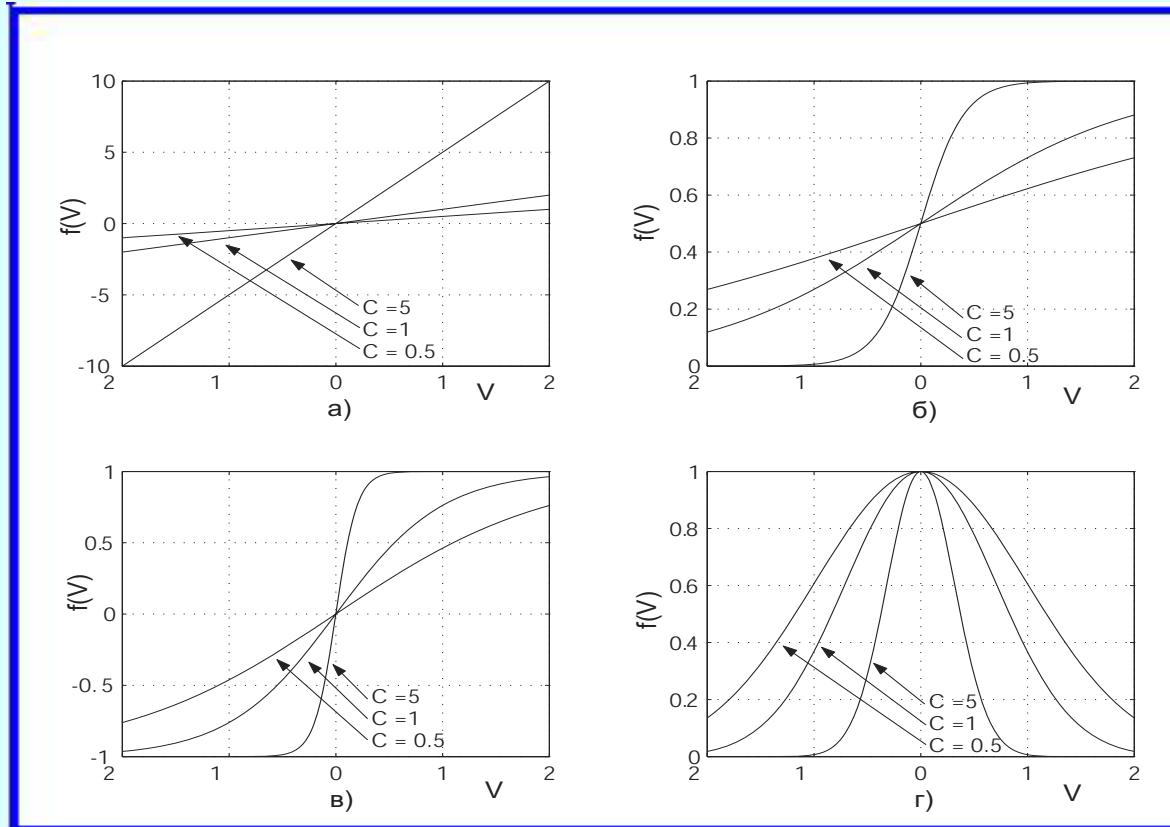
$$f(V) = \frac{e^{cV} - e^{-cV}}{e^{cV} + e^{-cV}} ;$$

- гауссовская функция

$$\varphi(V) = \exp\left(-\frac{V^2}{2c^2}\right) .$$

Основные структурные элементы НС-моделей (III)

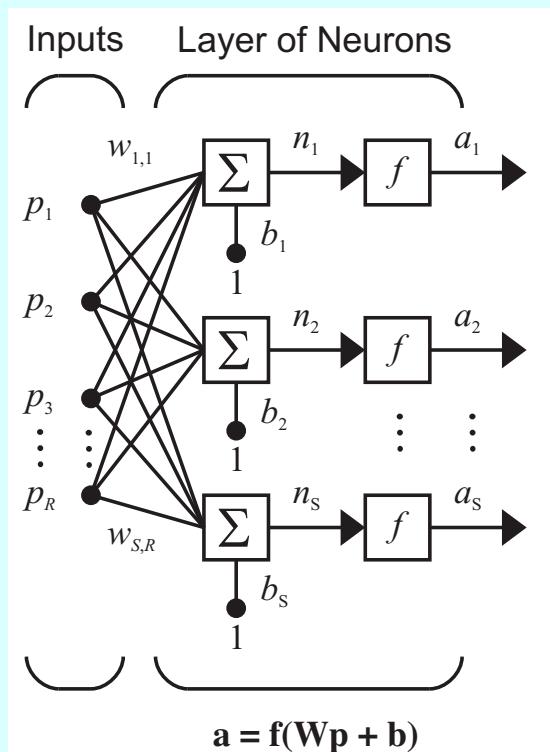
Активационные функции – 2:



- а** — линейная функция; **б** — логистическая функция (сигмоида);
- в** — гиперболический тангенс; **г** — гауссовская функция.

Основные структурные элементы НС-моделей (IV)

Слой нейронов – 1

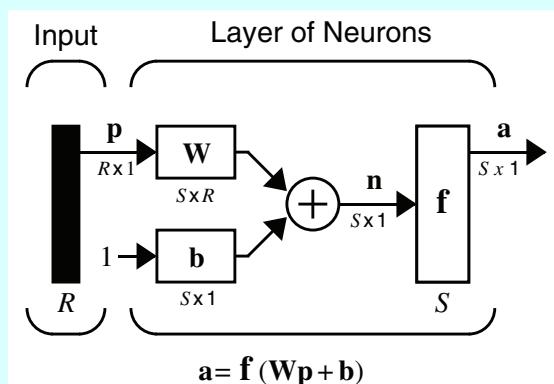


p_i — входы слоя,
 a_j — выходы слоя,
 $w_{k,i}$ — синаптические веса связей входов с нейронами слоя,
 b_k — смещения,
 $\Sigma(\cdot)$ — аддитивный сумматор,
 $f(\cdot)$ — активационная функция

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Fig.2-7, p.2-9).

Основные структурные элементы НС-моделей (V)

Слой нейронов – 2

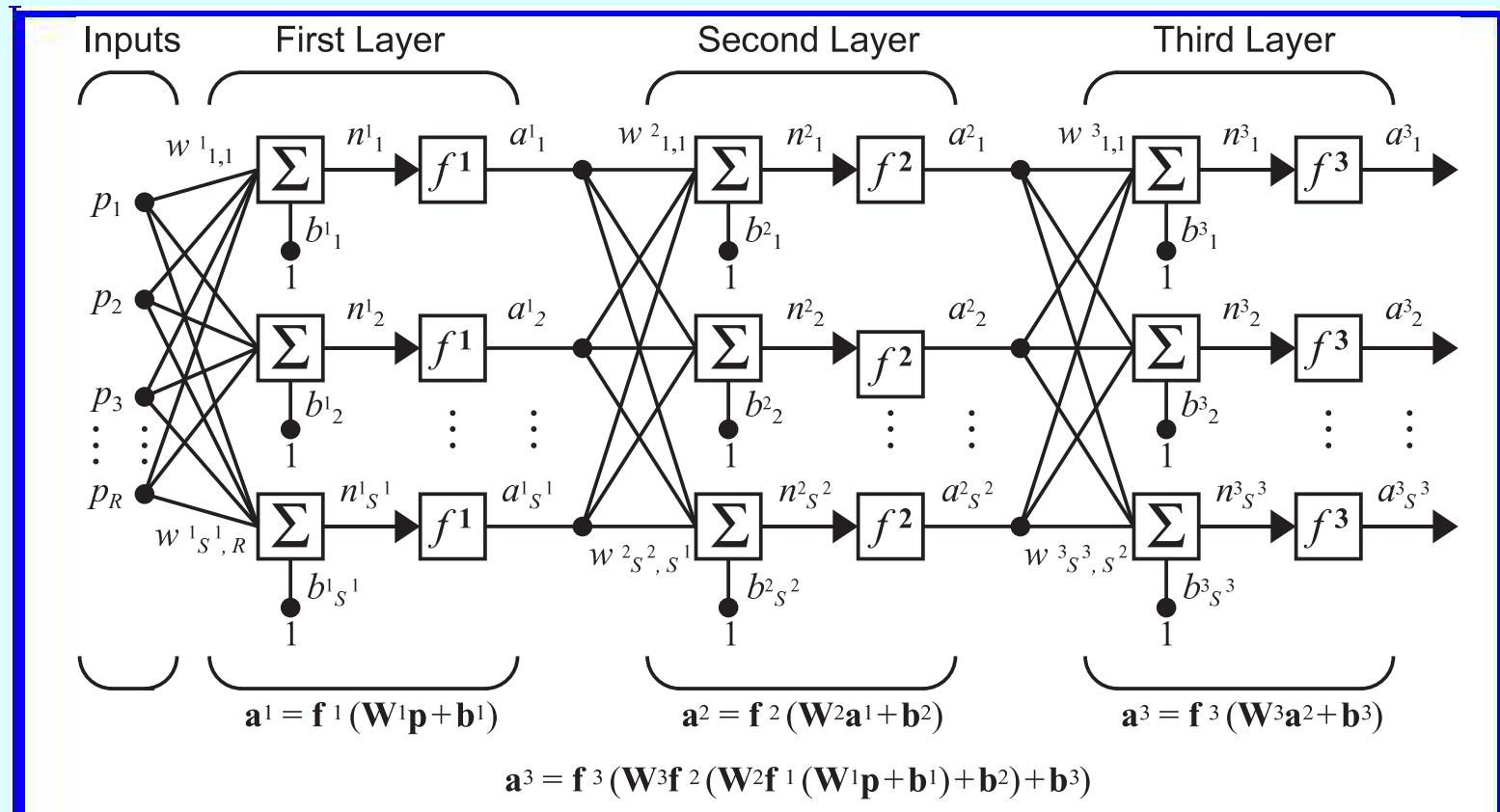


p — вектор входов слоя,
a — вектор выходов слоя,
W — матрица синаптических весов связей входов с нейронами слоя,
b — вектор смещений,
 $\Sigma(\cdot)$ — адаптивный сумматор,
 $f(\cdot)$ — активационная вектор-функция

Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Fig.2-8, p.2-10).

Основные структурные элементы НС-моделей (VI)

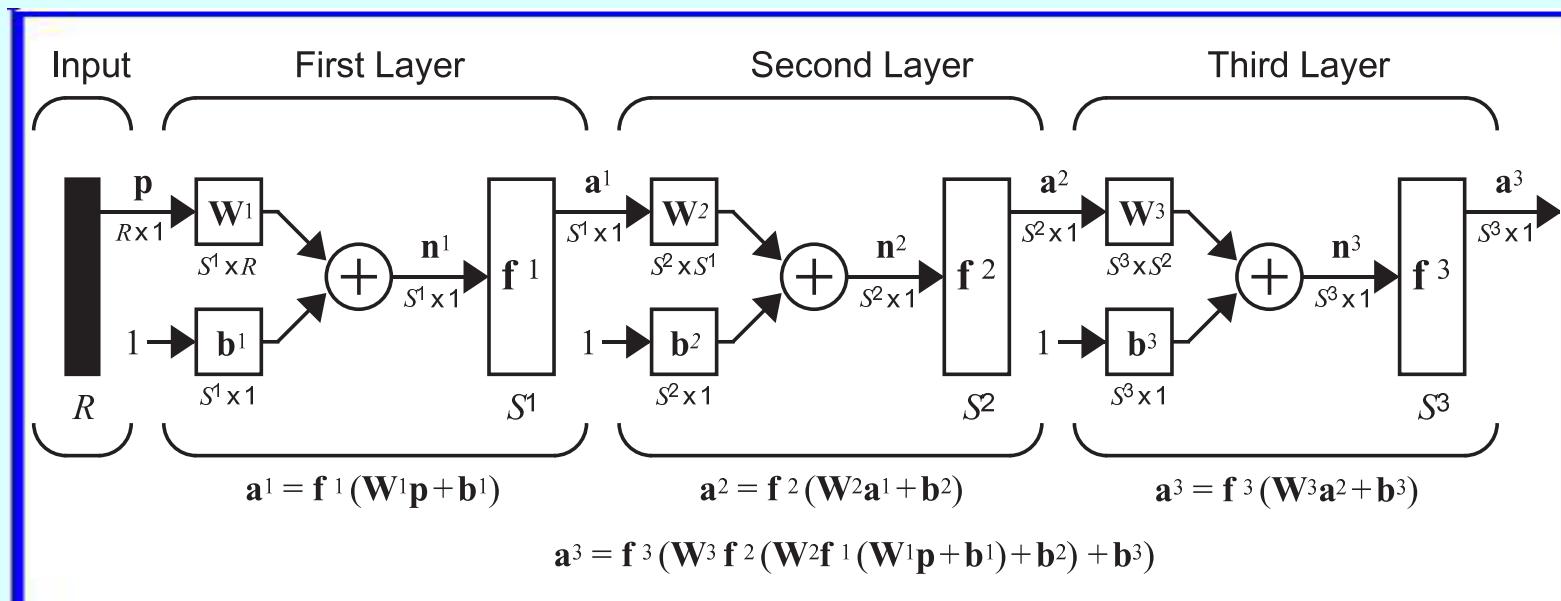
Трехслойная сеть – 1



Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Fig.2-9, p.2-11).

Основные структурные элементы НС-моделей (VII)

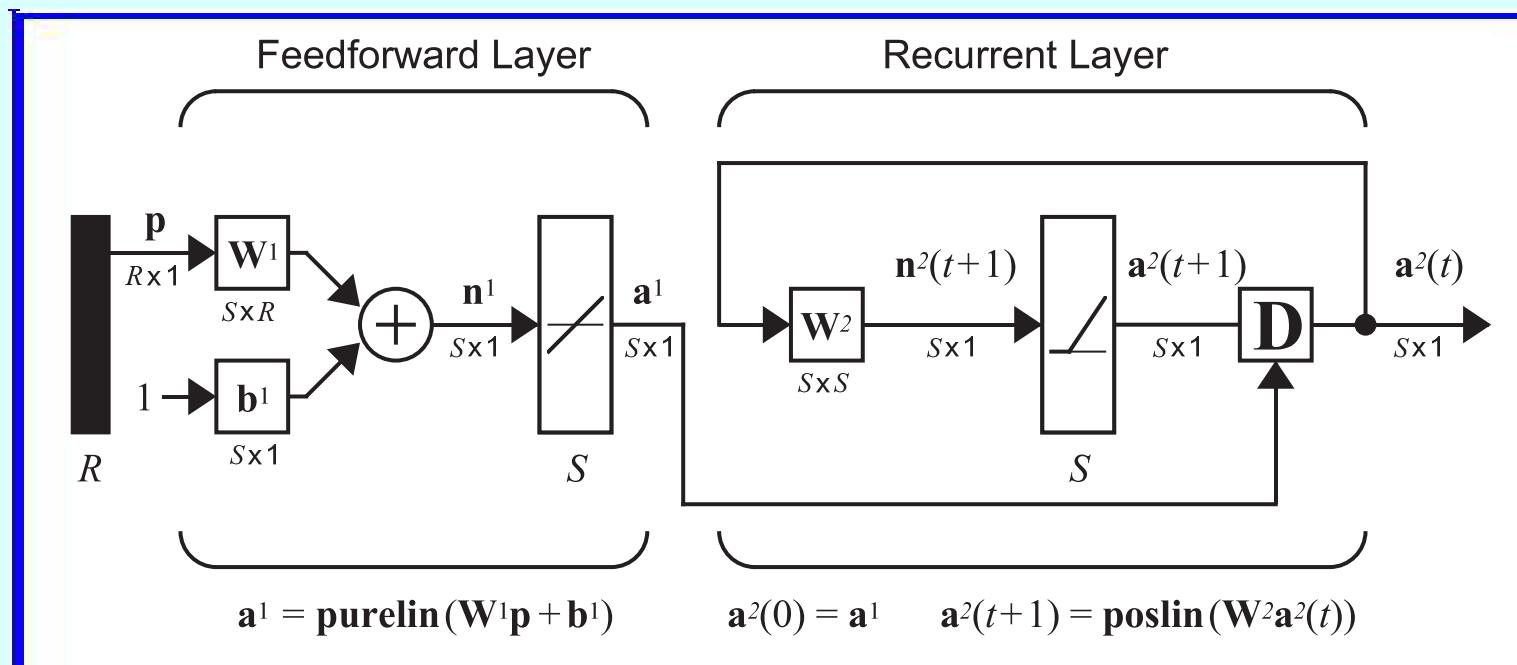
Трехслойная сеть – 2



Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Fig.2-10, p.2-12).

Основные структурные элементы НС-моделей (VIII)

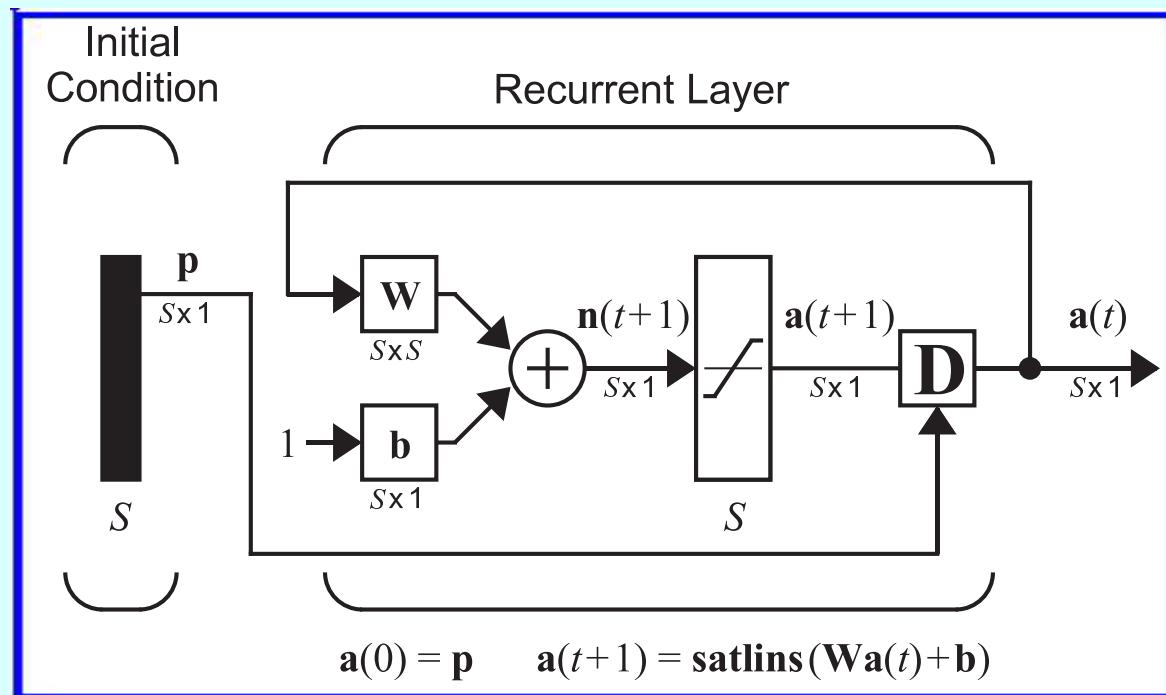
Пример динамической НС-модели — сеть Хемминга



Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Fig.3-5, p.3-8).

Основные структурные элементы НС-моделей (IX)

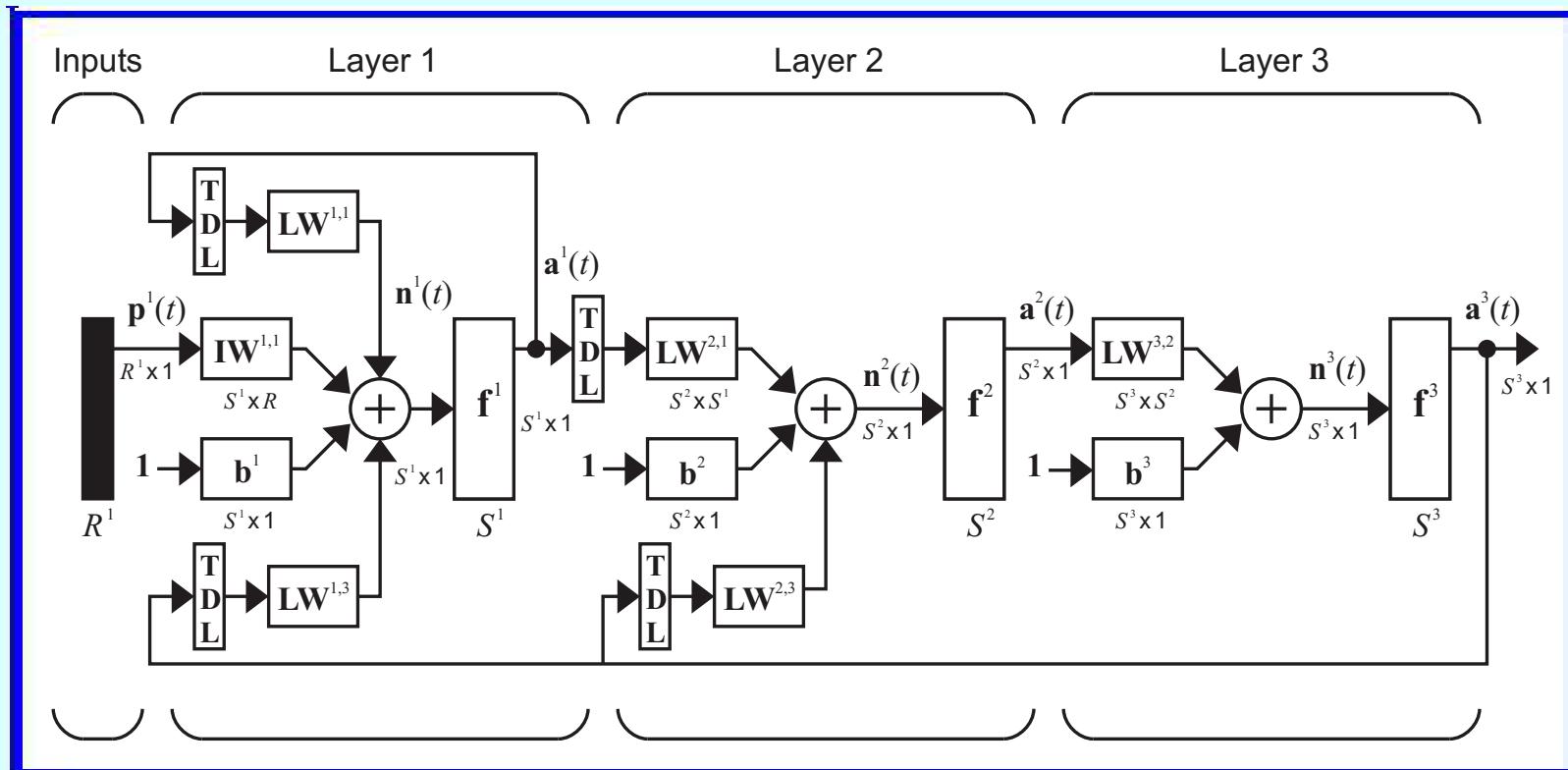
Пример динамической НС-модели — сеть Хопфилда



Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Fig.3-6, p.3-12).

Основные структурные элементы НС-моделей (Х)

Слоистая динамическая сеть (LDDN)

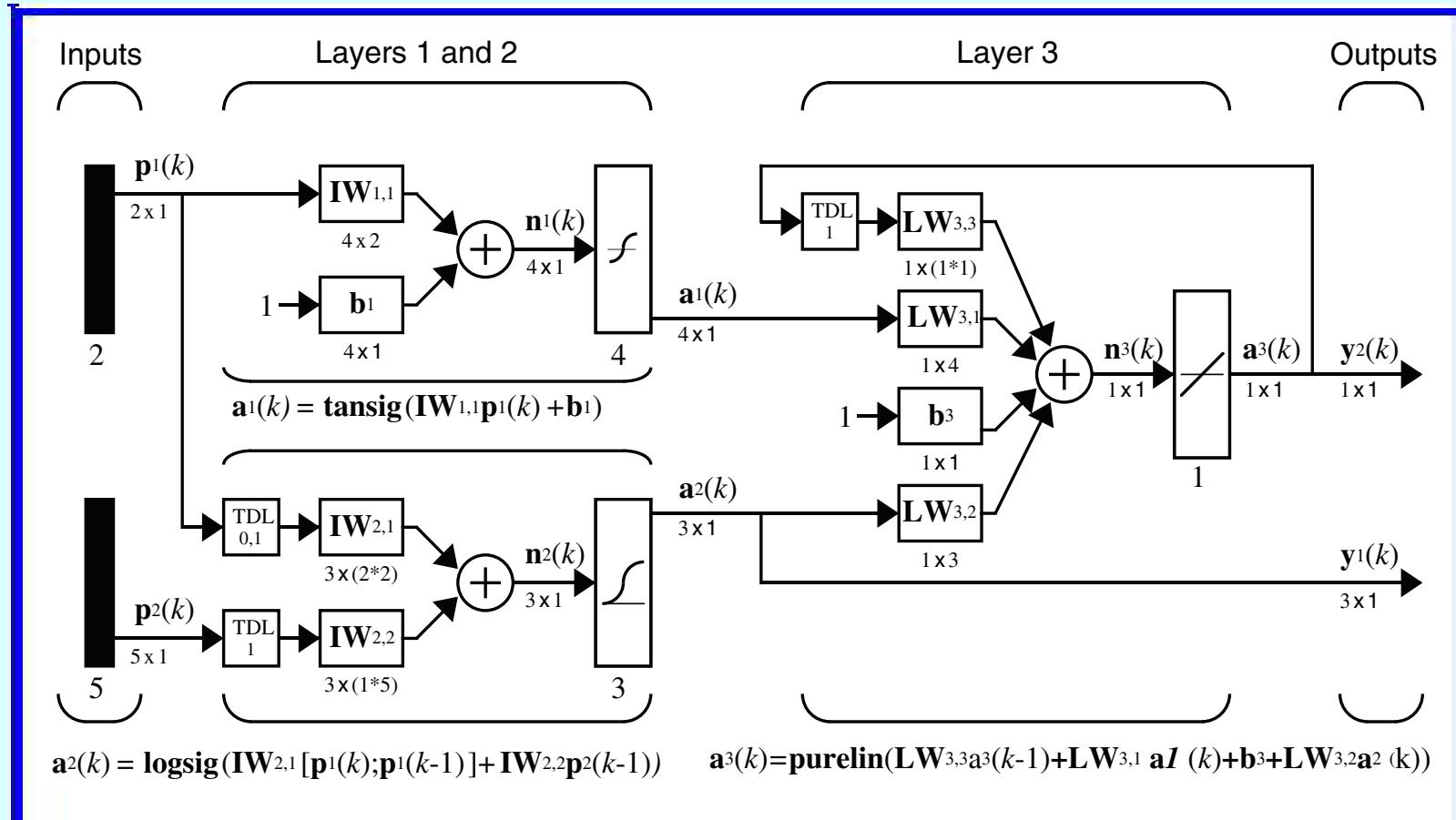


LDDN – Layered Digital Dynamic Network

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p.6-9).

Основные структурные элементы НС-моделей (XI)

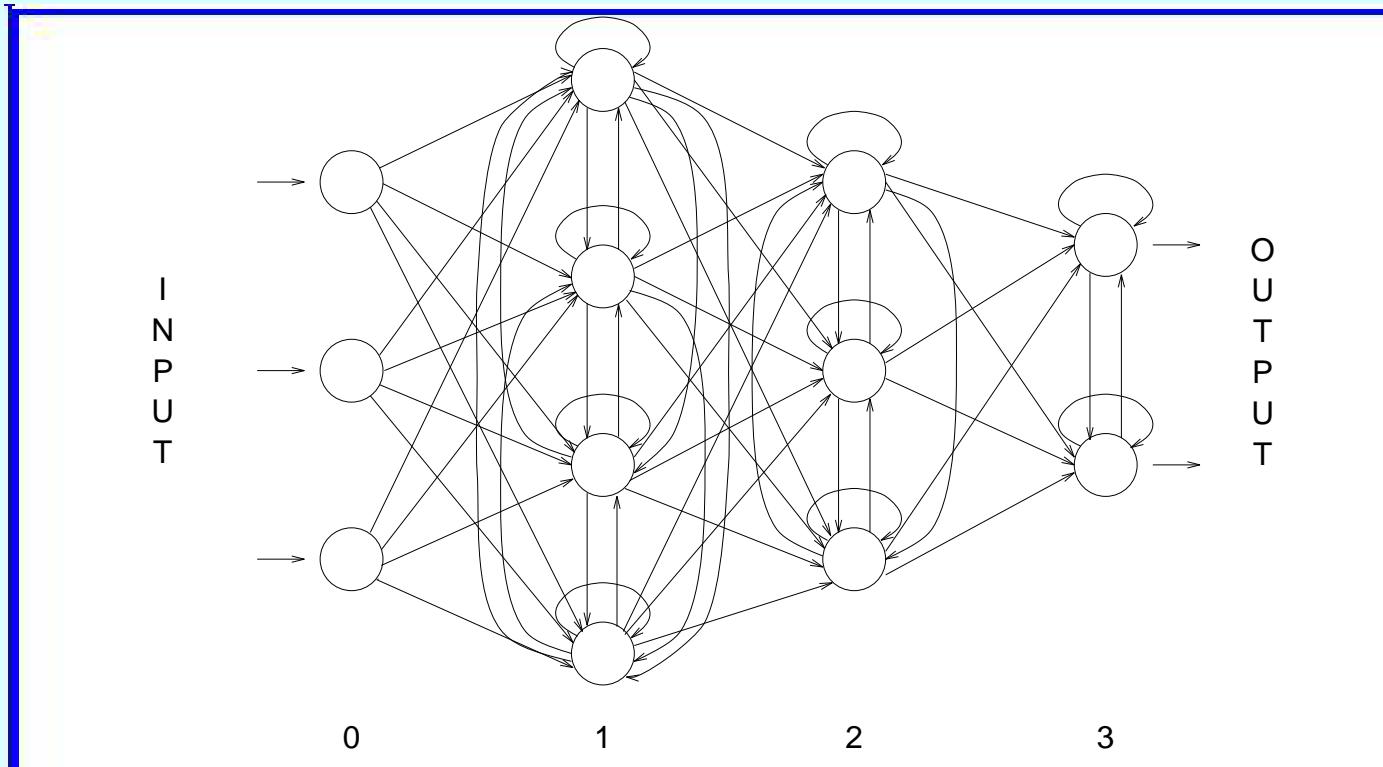
Комбинированная динамическая сеть



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p.12-3).

Основные структурные элементы НС-моделей (XII)

Слоистая НС-модель с латеральными связями

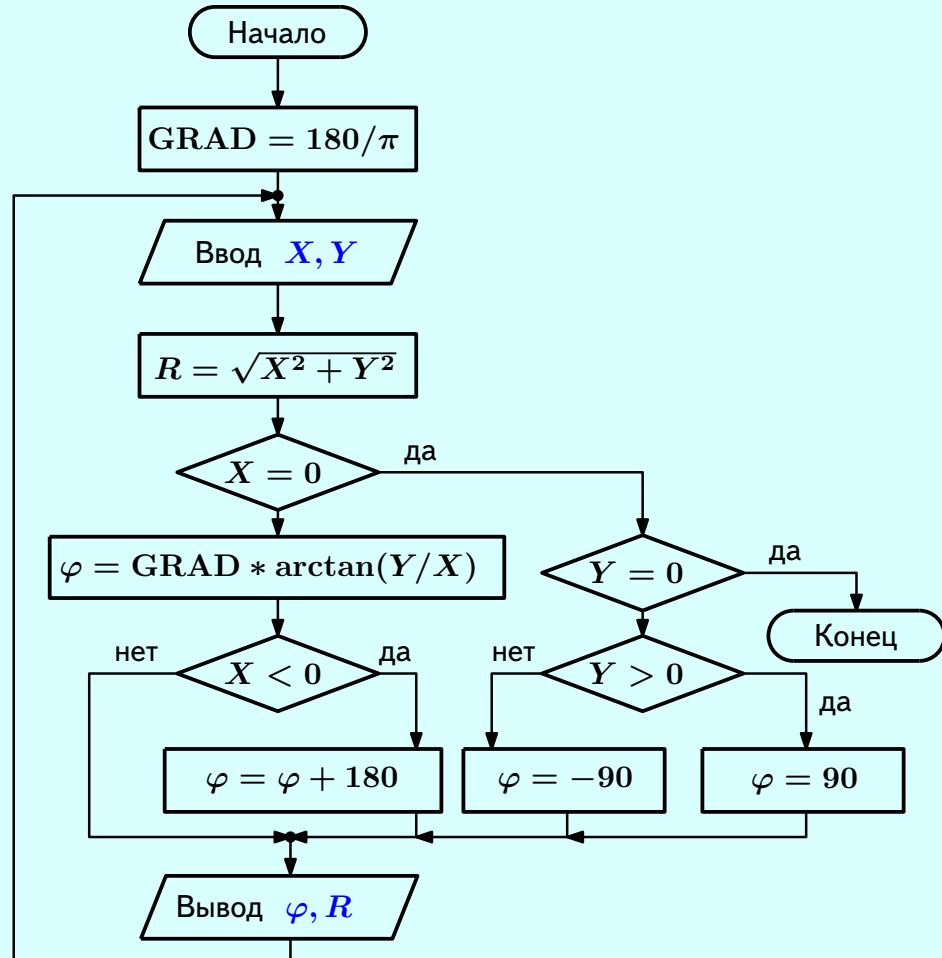


Рекуррентный мультиперсептрон (RMLP)

Источник: *Tutschku K.* Recurrent multilayer perceptrons for identification and control: The road to applications. – University of Würzburg, Institute of Computer Science, Research Report Series, Report No. 118, June 1995. – 28 pp. (Fig.4, p.8).

Основные структурные элементы НС-моделей (XIII)

Традиционная модель VS нейросетевая модель – 1



Преобразование прямоугольных координат в полярные:
традиционный алгоритм

X, Y — прямоугольные координаты,

φ, R — полярные координаты,

$$\varphi = \arctan(Y/X)$$

$$R = \sqrt{X^2 + Y^2}$$

Конец обработки:
 $X = Y = 0$

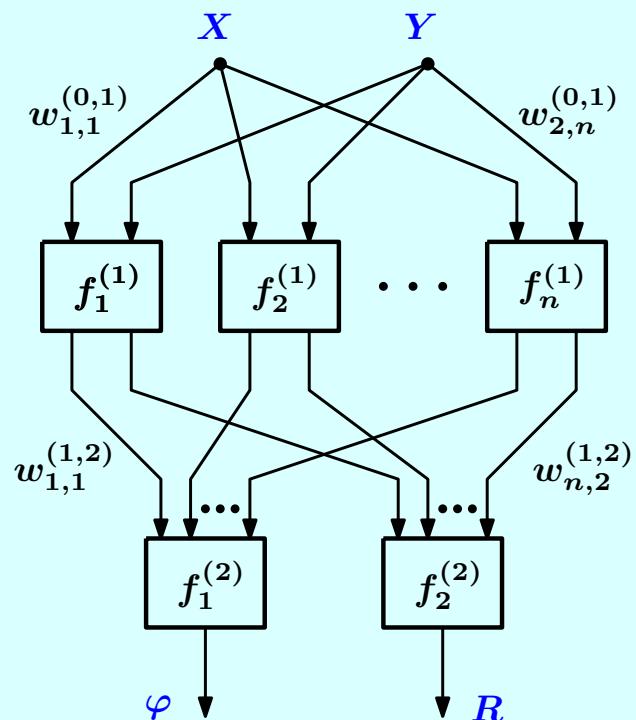
Выдавать главное значение арктангенса,
т.е. значение φ из диапазона

$$-\frac{\pi}{2} < \varphi < \frac{\pi}{2}$$

Источник: Браух В. Программирование на Фортране 77
для инженеров. – М.: Мир, 1987 (с.62, 174).

Основные структурные элементы НС-моделей (XIV)

Традиционная модель VS нейросетевая модель – 2



Преобразование прямоугольных координат в полярные:
нейросетевой алгоритм

X, Y — прямоугольные координаты,
 φ, R — полярные координаты,

Исходные данные (обучающий набор):
 $\langle X_i, Y_i \rangle \rightarrow \langle \varphi_i, R_i \rangle$,
 $i = 1, 2, \dots, m$

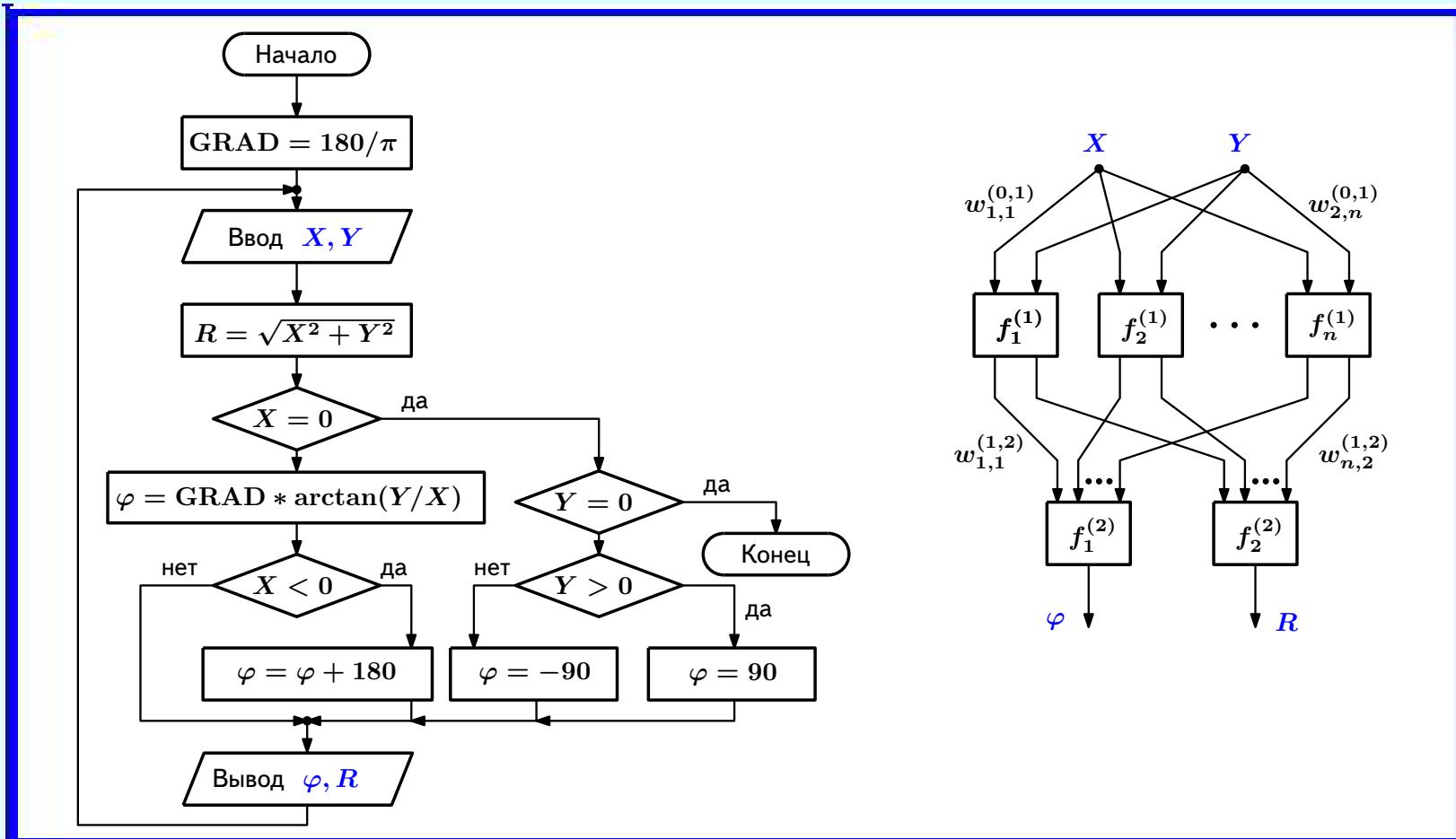
$f_j^{(1)}$, $j = 1, 2, \dots, n$ — композиция из сумматора и **нелинейной функции** одной переменной

$f_1^{(2)}, f_2^{(2)}$ — композиция из сумматора и **линейной функции** одной переменной

$w_{i,j}^{(0,1)}, w_{j,k}^{(1,2)}$ — варьируемые величины
(веса связей)

Основные структурные элементы НС-моделей (XV)

Традиционная модель VS нейросетевая модель – 3



Традиционная модель: сосредоточенная, специализированная
Нейросетевая модель: распределенная, универсальная

Типовые НС-задачи (I)

Типовые (характерные) задачи НС-моделирования:

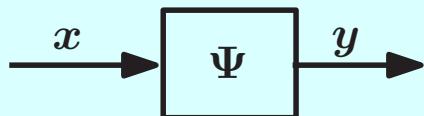
- представление зависимостей между величинами;
- классификация (категоризация) данных;
- выявление закономерностей в данных;
- кластеризация данных;
- сжатие данных;
- представление (многомерных) данных в пространствах меньшей размерности;
- ассоциативная память;
- оптимизация.

Искусство постановщика/аналитика состоит в том, чтобы:

- увидеть** такую типовую задачу или их комбинацию в решаемой прикладной проблеме;
- выразить** прикладную проблему в терминах типовых задач;
- решить** прикладную проблему, используя методы, имеющиеся для соответствующих типовых задач.

Типовые НС-задачи (II)

Представление зависимостей между величинами – 1



$$x = (x_1, \dots, x_n)$$

$$y = (y_1, \dots, y_m)$$

$$x_i, y_j \in \mathbb{R}, \mathbb{C}, \mathbb{Z}, \dots$$

$$x_i, y_j \in \mathcal{C}[a, b], \mathcal{C}^k[a, b], \mathcal{L}[a, b], \dots$$

$\mathbb{R}, \mathbb{C}, \mathbb{Z}$ — множества действительных, комплексных и целых чисел, соответственно.

$\mathcal{C}[a, b], \mathcal{C}^k[a, b], \mathcal{L}[a, b]$ — множества функций на отрезке $[a, b]$ (непрерывных, k раз дифференцируемых и интегрируемых, соответственно).

Ψ — отображение (статическое или динамическое).

Возможные комбинации значений входных (x_i , $i = 1, \dots, n$) и выходных (y_j , $j = 1, \dots, m$) величин для отображения Ψ :

1. x_i, y_j — числа ($y = \Psi(x)$, функция);
2. x_i — числа, y_j — кривые ($\dot{y} = \Phi(y, t)$, $y(t_0) = y_0 = x$, неуправляемая динамическая система);
3. x_i, y_j — кривые ($\dot{y} = \Phi(y, u, t)$, $y(t_0) = y_0 = x$, $u \in U$, управляемая динамическая система);
4. x_i — кривые, y_j — числа ($y = \int_a^b \Phi(x(t)) dt$, функционал).

Методы и средства нейроинформатики позволяют работать как со статическими, так и с динамическими зависимостями (моделями).

Типовые НС-задачи (III)

Представление зависимостей между величинами – 2

Статические и динамические зависимости (модели):

	Статические модели	Динамические модели
Прогон модели	Однократный	Многократный
Внутр. состояния	Нет	Есть

Примеры статических моделей:

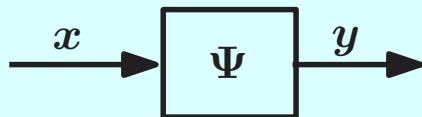
- 1) «обычная» функция $y = f(x)$, $x, y \in \mathbb{R}^1$
- 2) отображение $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Примеры динамических моделей:

- 1) дифференциальное уравнение $\frac{dy}{dt} = f(y, t)$
- 2) рекуррентная формула $y_{n+1} = y_n + F(y_n, t_n)$, $y_n = y(t_n)$

Типовые НС-задачи (IV)

Представление зависимостей между величинами – 3



$$x = (x_1, \dots, x_n)$$

$$y = (y_1, \dots, y_m)$$

$$x_i, y_j \in \mathbb{R}, \mathbb{C}, \mathbb{Z}, \dots$$

$$x_i, y_j \in \mathcal{C}[a, b], \mathcal{C}^k[a, b], \mathcal{L}[a, b], \dots$$

$\mathbb{R}, \mathbb{C}, \mathbb{Z}$ – множества действительных, комплексных и целых **чисел**, соответственно.

$\mathcal{C}[a, b], \mathcal{C}^k[a, b], \mathcal{L}[a, b]$ – множества **функций** на отрезке $[a, b]$ (непрерывных, k раз дифференцируемых и интегрируемых, соответственно).

Ψ – **отображение** (статическое или динамическое).

Как получить Ψ ?

Возможные варианты действий:

- 1) «угадать» (алгоритмизация);
- 2) «подогнать под ответ» (идентификация).

«Угадать»: имеется *интуитивное представление* о том, что должно получиться, на его основе надо вывести требуемое соотношение *теоретическим путем* и выразить в виде соответствующего алгоритма (как правило, *вручную!*).

«Подогнать под ответ»: имеются *экспериментальные данные*, требуется найти приближенное соотношение, которому удовлетворяют эти данные.

Нейроинформатика имеет дело со **вторым** из этих вариантов действий.

Типовые НС-задачи (V)

Представление зависимостей между величинами – 4

Как получить Ψ ?

Возможные подходы:

- 1) **теоретическая** модель, получаемая «из первых принципов» и/или на основе других теоретических моделей (**white box**);
- 2) **эмпирическая** модель, получаемая исключительно на основе экспериментальных данных (**black box**);
- 3) **полуэмпирическая** модель, получаемая с привлечением как теоретического знания, так и экспериментальных данных (**gray box**).

Типовые НС-задачи (VI)

Классификация данных – 1

- Пусть имеется счетное множество X объектов $x_i \in X$,
 $i = 1, 2, \dots, n, \dots$
- Множество X разделено на совокупность непересекающихся подмножеств (классов) X_k , $k = 1, 2, \dots, s$:

$$X_1 \cup X_2 \cup \dots \cup X_k \cup \dots \cup X_s = X$$

$$X_1 \cap X_2 \cap \dots \cap X_k \cap \dots \cap X_s = \emptyset$$

Проблема. Для произвольного объекта $x_i \in X$ определить, какому классу X_k , $k \in K$, $K = \{1, 2, \dots, s\}$ он принадлежит, т.е. необходимо построить и использовать отображение $\Psi : X \rightarrow K$.

Пример. Объекты x_i — **буквы** некоторого алфавита \mathcal{A} , каждая из них во всевозможных вариантах начертаний; множество X — совокупность всевозможных **вариантов начертаний всех букв** алфавита \mathcal{A} ; X_k — всевозможные варианты начертаний для k -й буквы алфавита \mathcal{A} .

Проблема — распознавание графических символов (**OCR** — Optical Character Recognition), Ψ — программа распознавания.

Типовые НС-задачи (VII)

Классификация данных – 2

**Источник разнообразия классифицируемых объектов
(на примере задачи OCR):**

- ❑ Печатный шрифт (алфавит \mathcal{A}) существует в виде большого числа разновидностей-гарнитур (например, Times New Roman, Courier, Arial и т.п.).
- ❑ Буквы могут пропечатываться не идеально, а с какими-то дефектами.
- ❑ Рукописные варианты буквы очень сильно варьируются по написанию.

Потенциально возможное **число вариантов**: первый пункт — *конечное*, второй и третий — *бесконечное*.

Объекты $x_i \in X$ должны быть **отделимы** друг от друга, т.е. они должны **различаться** по каким-либо признакам.

Типовые НС-задачи (VIII)

Классификация данных – 3

Основные проблемы, связанные с решением задач классификации:

- Проблемы, связанные с **методами решения** задач классификации.
- Проблемы, связанные со **спецификой данных**, для которых решается задача классификации.

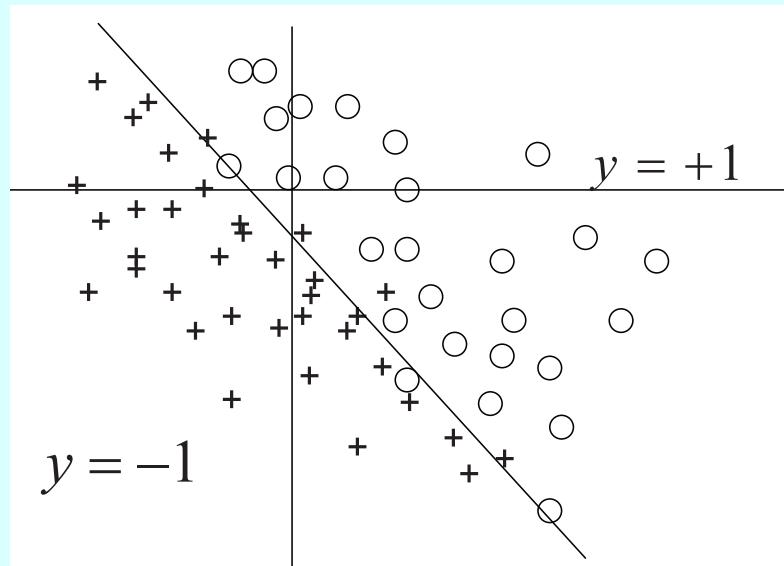
Методы решения — как описывать объекты, как их сопоставлять между собой, предобработка данных.

Специфика данных — невозможность однозначной классификации некоторых объектов; сложный характер границ между классами.

Типовые НС-задачи (IX)

Классификация данных – 4

Проблема представления объектов классификации (1)



Случай, когда объект $x_i \in X$ можно охарактеризовать набором из N **признаков**, т.е. параметров, принимающих **числовые значения** (размеры, масса и т.п.).

Признаки $w_j \in W_j$, $j = 1, 2, \dots, N$.

Вектор признаков $w = (w_1, w_2, \dots, w_N)$.

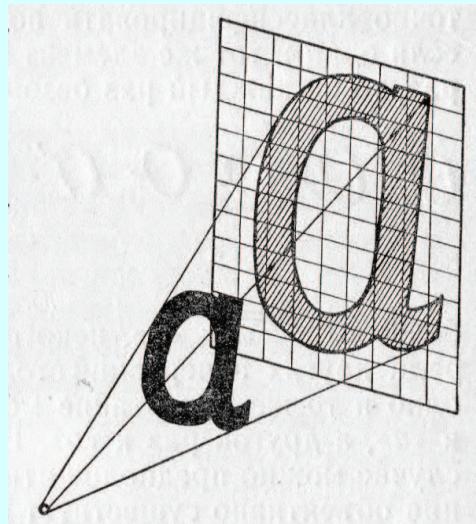
Признаковое пространство $W = W_1 \times W_2 \times \dots \times W_N$ как совокупность всех возможных комбинаций значений (декартово произведение) признаков.

Объект полностью характеризуется N -кой $\langle w_1, w_2, \dots, w_N \rangle \in W$.

Типовые НС-задачи (Х)

Классификация данных – 5

Проблема представления объектов классификации (2)



Отображение объекта реального мира на рецептивную среду (сетчатка глаза, матрица фотокамеры и т.п.)

Терминология:

Классы объектов ≡ **Образы**

Объекты, принадлежащие классу ≡ **Паттерны**

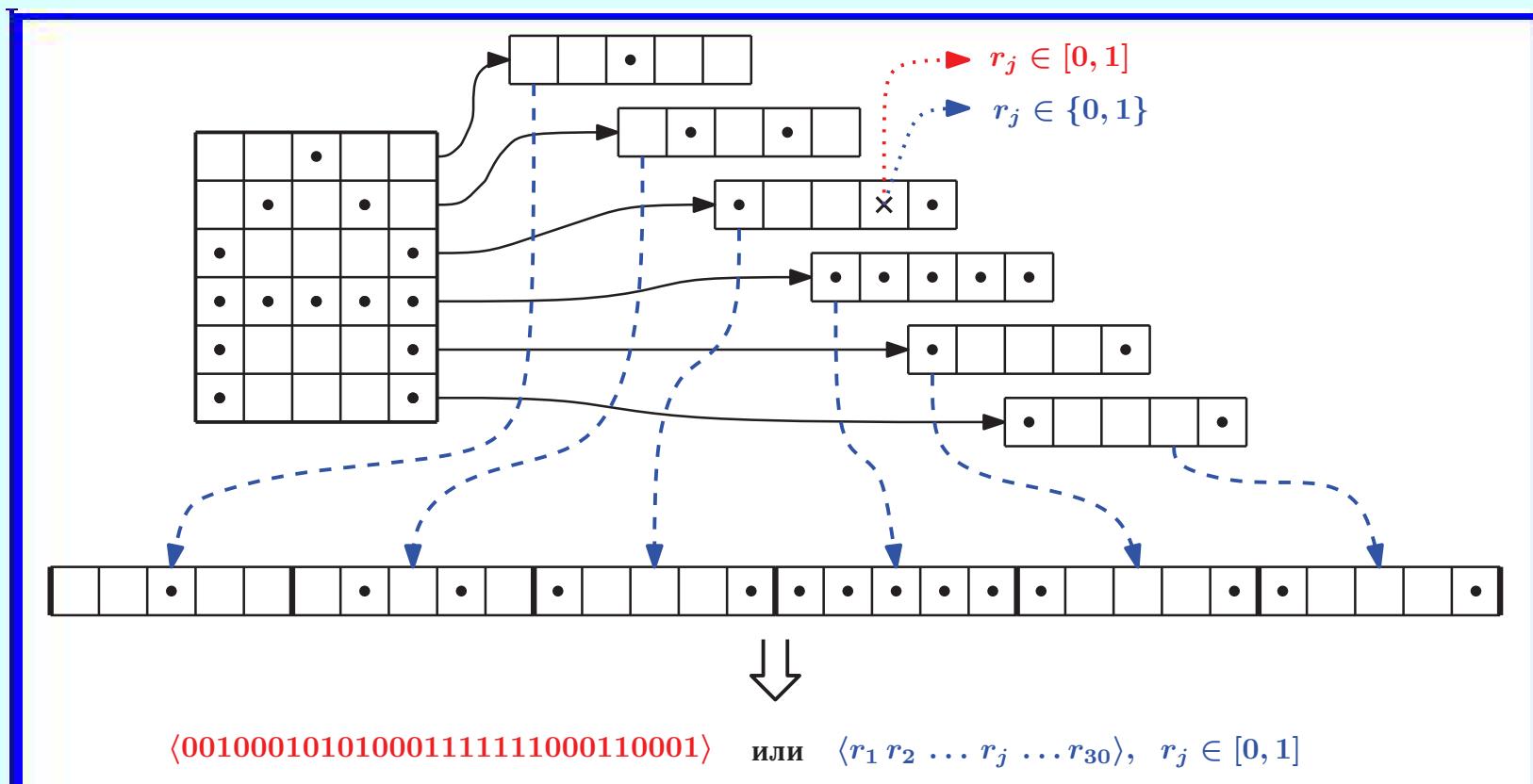
Задача классификации ≡ Задача **распознавания образов**

Источник: Айзerman M.A., Браверман Э.М., Розоноэр Л.И. Метод потенциальных функций в теории обучения машин. – М.: Наука, 1970. – 384 с. (рис.1, с.13)

Типовые НС-задачи (XI)

Классификация данных – 6

Проблема представления объектов классификации (3)



Типовые НС-задачи (XII)

Классификация данных – 7

Проблема вариативности классифицируемых данных (1)

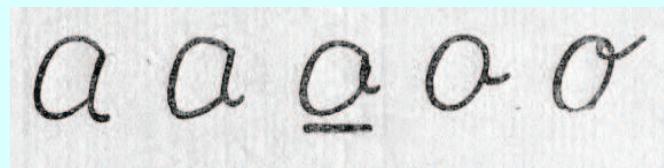
65473	60198	68544
<u>70065</u>	70117	<u>19032</u> 96720
27260	61828	,9559
74136	<u>1932</u>	63101
20878	60521	38002
48640-2398	<u>20907</u>	14868

Примеры рукописной записи почтовых кодов, взятые из тестовой базы данных Почтовой службы США (Из: Prévotet J.-C. Tutorial on Neural Networks, Slide 18).

Типовые НС-задачи (XIII)

Классификация данных – 8

Проблема вариативности классифицируемых данных (2)



Верхний рисунок:

Первая буква — классифицируется как «**а**» без сомнений.

Вторая буква — классифицируется как «**а**» с сомнениями.

Третья буква — не может быть классифицирована как «**а**» или «**о**».

Четвертая буква — классифицируется как «**о**» с сомнениями.

Пятая буква — классифицируется как «**о**» без сомнений.

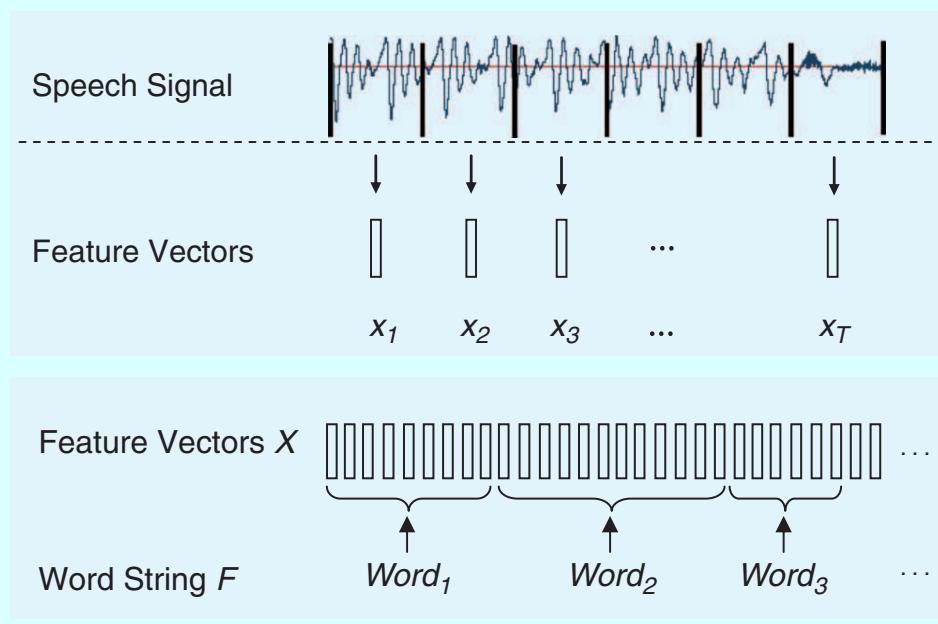
См.: Айзerman М.А., Браверман Э.М., Розоноэр Л.И. Метод потенциальных функций в теории обучения машин. – М.: Наука, 1970. – 384 с. (рис.2, с.14)

Нижний рисунок — пример, демонстрирующий уровень возможностей систем автоматического распознавания текстов.

Типовые НС-задачи (XIV)

Классификация данных – 9

Проблема вариативности классифицируемых данных (3)



Проблема распознавания речи

Процесс **параметризации речевого сигнала**, представляемого как последовательность векторов результатов наблюдения (*верхний рисунок*).

Результат этого процесса — набор признаковых векторов (feature vectors) $X_i, i = 1, \dots, T$.

Процесс **распознавания речевого сигнала**, представленного как последовательность слов (символьных строк), объединяющих фрагменты набора признаковых векторов (*нижний рисунок*).

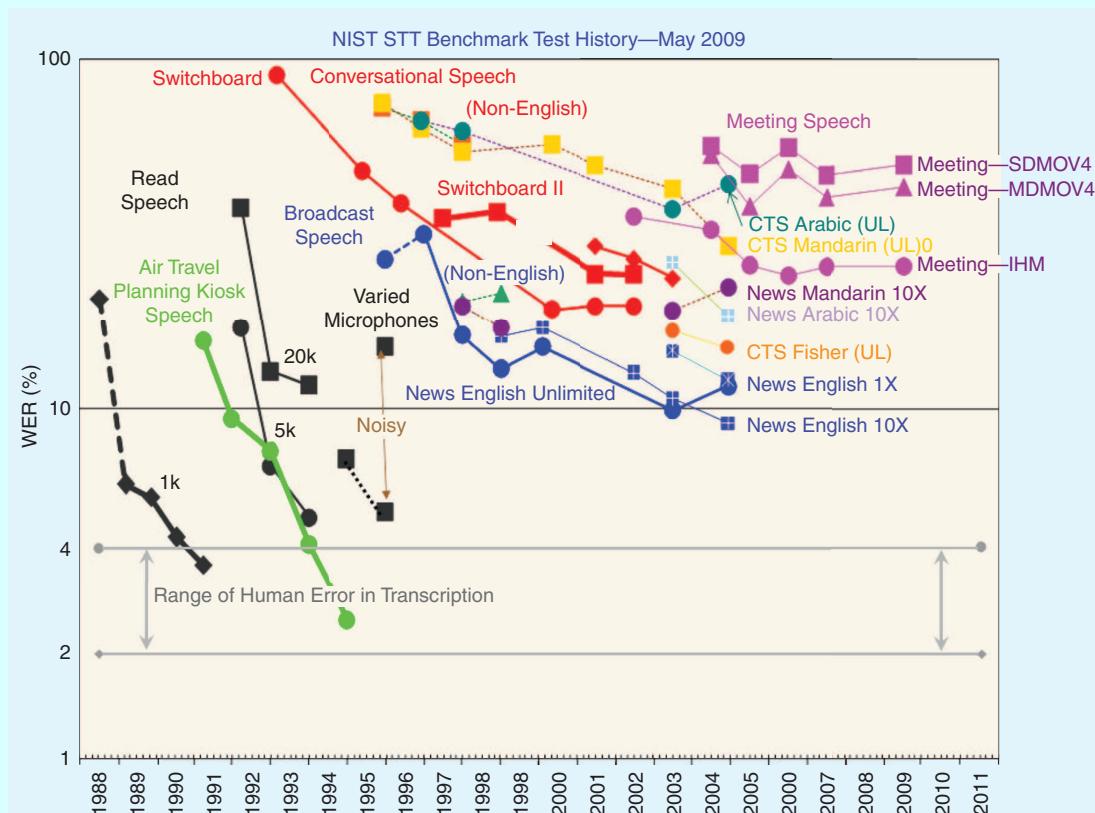
Распознаватель речи работает с полученной последовательностью слов (word strings).

Источник: *He X., Deng L. Speech recognition, machine translation, and speech translation: A unified discriminative learning paradigm // IEEE Signal Processing Magazine. – Sept. 2011. – pp. 126–133.*

Типовые НС-задачи (XV)

Классификация данных – 10

Проблема вариативности классифицируемых данных (4)



Результаты решения тестовых задач по **автоматическому распознаванию речи** с использованием тестовой базы данных NIST

NIST — National Institute of Standards and Technology

STT — Speech-To-Text

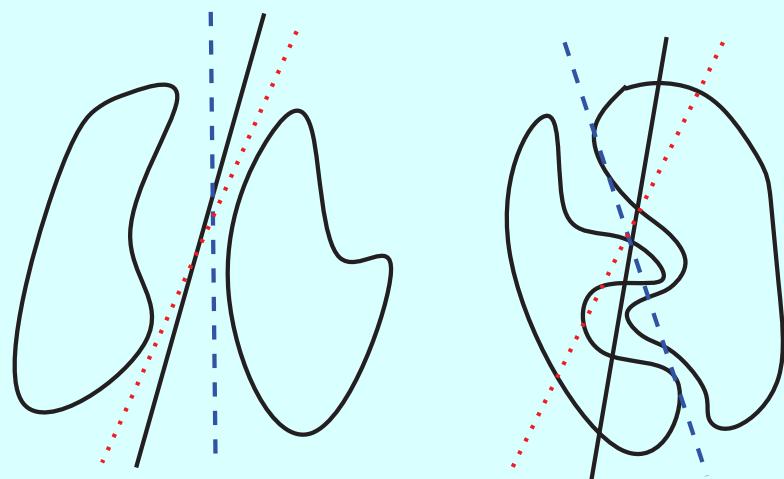
WER — Word Error Rate

Источник: *He X., Deng L.*. Speech recognition, machine translation, and speech translation: A unified discriminative learning paradigm // *IEEE Signal Processing Magazine*. – Sept. 2011. – pp. 126–133.

Типовые НС-задачи (XVI)

Классификация данных – 11

Проблема разделимости классов (1)



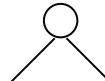
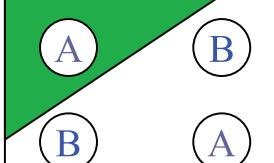
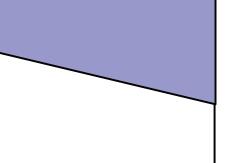
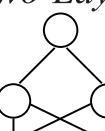
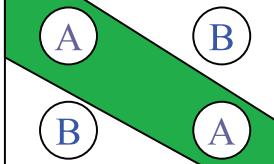
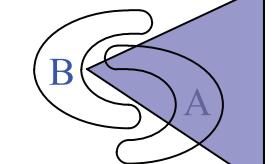
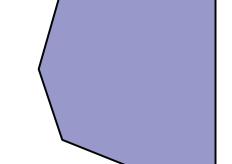
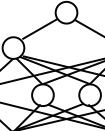
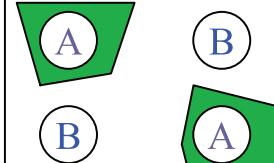
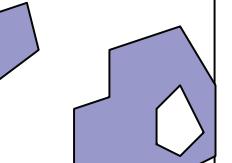
Левый рисунок: Классы (области возможных значений признаков для них) компактны и **линейно разделимы**.

Правый рисунок: Классы **не являются** компактными и линейно разделимыми.

Типовые НС-задачи (XVII)

Классификация данных – 12

Проблема разделимости классов (2)

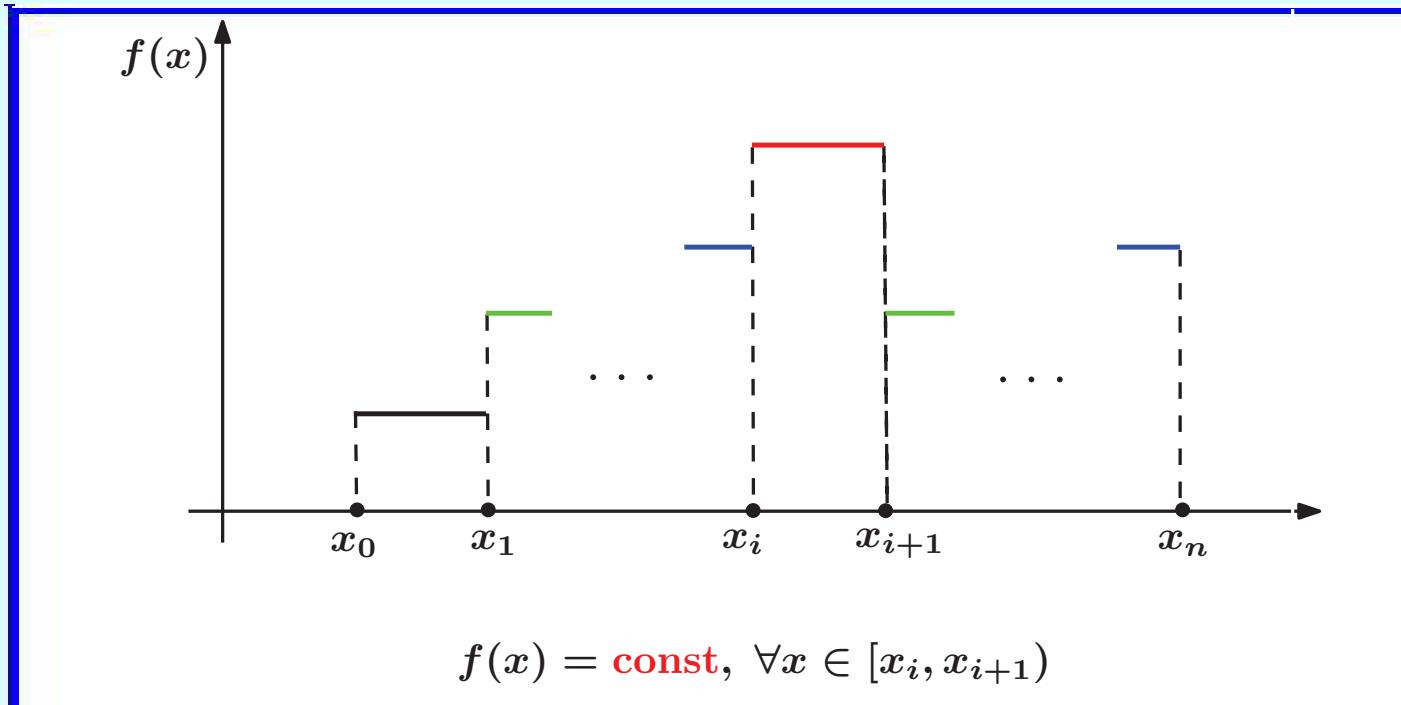
Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

Источник: [Lippman R.P.](#) An introduction to computing with neural net // Acoustics, Speech and Signal Processing Magazine. – April 1987. – Vol 3, No. 4. – pp.4–22 (Figure 14, p. 14).

Типовые НС-задачи (XVIII)

Классификация данных – 13

Связь задач классификации и аппроксимации



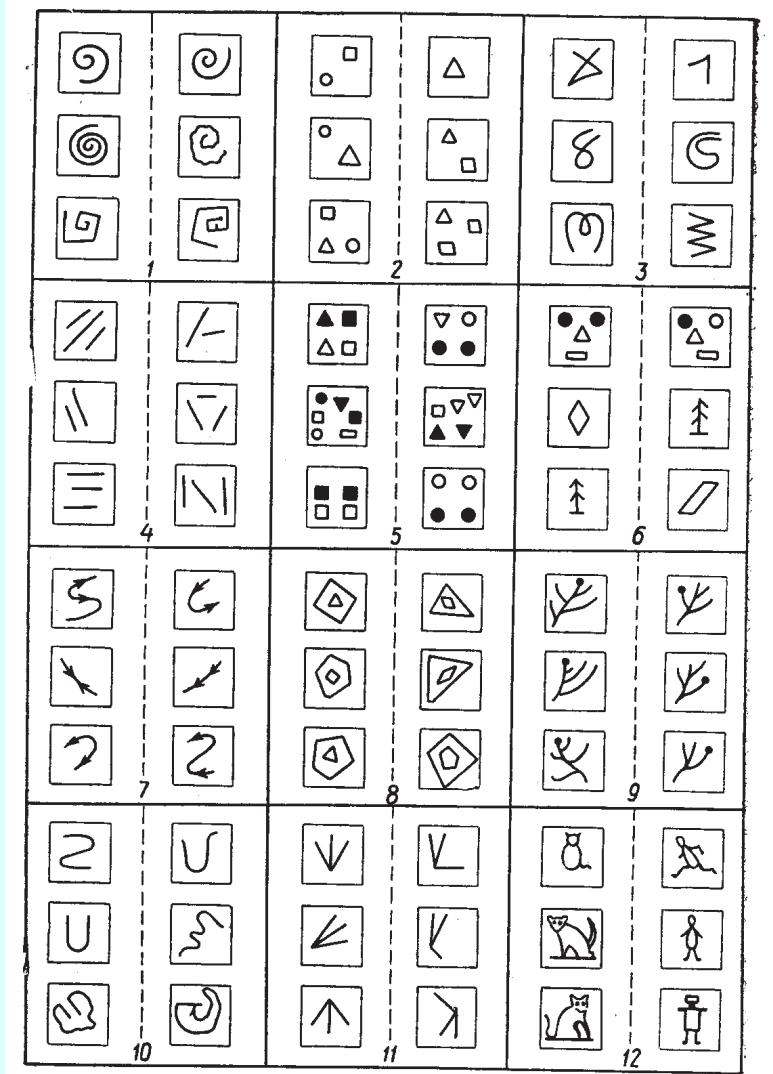
Задача классификации как задача нахождения **кусочно-постоянной** функции.

Значение функции $f(x), \forall x \in [x_i, x_{i+1})$ — **метка класса**,
объединяющего элементы $x \in [x_i, x_{i+1})$.

Класс **не обязательно** занимает **односвязную область** в признаковом пространстве.

Типовые НС-задачи (XIX)

Выявление закономерностей в данных – 1



Проблема узнавания

Формирование **вектора признаков** как **творческая задача**, требующая абстрактного и логического мышления.

12 из 100 задач, иллюстрирующих проблему узнавания ([Бонгард М.М. Проблема узнавания. – М.: Наука, 1967. – 320 с.](#))

Каждая из задач — шесть объектов: по три из двух классов (левый и правый столбец, соответственно).

Требуемое решение: *разгадать* принцип разделения объектов на классы (т.е. найти *существенные признаки*, отличающие картинки из разных классов) и *дополнить* классы новыми объектами (т.е. обеспечить *неограниченную расширяемость* классов).

Источник: [Васильев В.И. Проблемы обучения распознаванию образов: Принципы, алгоритмы, реализация. – К.: Выща школа, 1989. – 64 с.](#)

Типовые НС-задачи (ХХ)

Выявление закономерностей в данных – 2

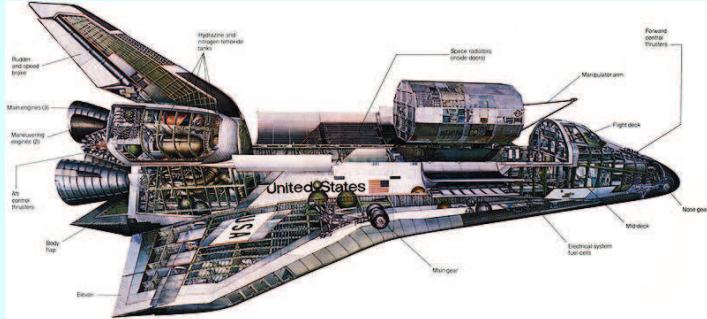
Проблема узнавания — ответы

	Правило для левого столбца	Правило для правого столбца
1	Сpirаль закручивается влево	Сpirаль закручивается вправо
2	Есть круг	Нет круга
3	Линия имеет самопересечения	Линия не имеет самопересечений
4	Отрезки почти параллельны	Отрезки расположены под большим углом
5	Черные фигуры выше белых	Белые фигуры выше черных
6	Есть симметрия	Нет симметрии
7	Стрелки направлены в разные стороны	Стрелки направлены в одну сторону
8	У внешней фигуры больше углов	У внешней фигуры меньше углов
9	Груз на главной ветке	Груз на боковой ветке
10	Концы кривой параллельны	Концы кривой перпендикулярны
11	Угол разделен пополам	Угол разделен не пополам
12	Кошка	Человечек

Источник: *Васильев В.И.* Проблемы обучения распознаванию образов: Принципы, алгоритмы, реализация. – К.: Выща школа, 1989. – 64 с. (таблица на с.10)

Типовые НС-задачи (ХXI)

Выявление закономерностей в данных – 3



Стендовые испытания (наземные) SSME:

Space Shuttle — 5 шаттлов, 135 полетов (1981–2011).

SSME — Space Shuttle Main Engine

Общее число огневых испытаний — **свыше 1200** (за 1975–1980 гг.).

Число измеряемых и регистрируемых величин — **от 300 до 500**.

Число величин, используемых для контроля процесса испытаний и управления им — **свыше 100**.

Автоматизированная **система контроля** хода испытаний для **выявления аномалий** в поведении SSME и аварийного **прекращения эксперимента** — комплекс средств измерений и принятия решений (автоматика + человек-оператор). **Автоматика**: схемы голосования и проверки различных условий.

Число запусков, завершившихся **аварийно** с повреждением испытательного стенда — **27**.

Альтернативный вариант (с 1988 г.) — выявление в реальном времени закономерностей в изменениях совокупности регистрируемых величин, приводящих к аварийному исходу. Удается обнаружить возникновение аварийной ситуации на ранней стадии.

Типовые НС-задачи (XXII)

Кластеризация данных – 1

Два варианта исходных данных для построения НС-модели:

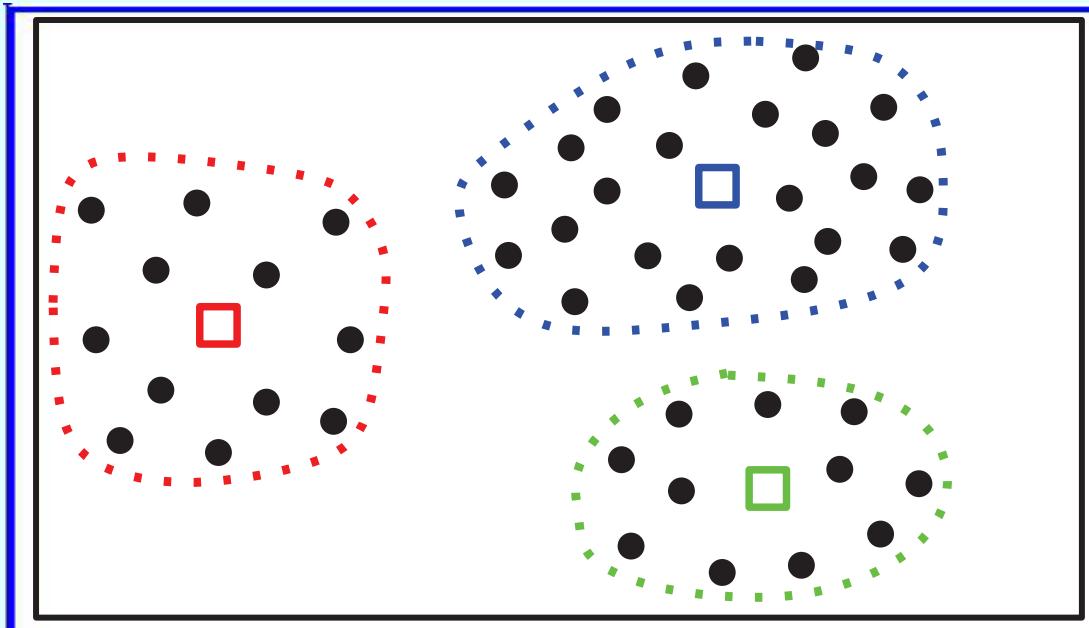
- набор $\{\mathbf{x}_i, f(\mathbf{x}_i)\}, i = 1, \dots, N_p$;
- набор $\{\mathbf{x}_i\}, i = 1, \dots, N_p$.

В **первом варианте** исходных данных для каждого \mathbf{x}_i *присутствует в явном виде* требуемое значение формируемой функции $f(\mathbf{x}_i)$ или указание на метку класса, к которому принадлежит \mathbf{x}_i .

Во **втором варианте** имеются только входные данные $\{\mathbf{x}_i\}$, отвечающие им выходные данные $\{f(\mathbf{x}_i)\}$ *отсутствуют*.

Типовые НС-задачи (ХХIII)

Кластеризация данных – 2



Для классификации элементов $\{x_i\}$ из входного набора данных в отсутствие данных о метках классов для этих элементов — **информация о взаимном расположении** для $\{x_i\}$.

Результат — группировка элементов входного набора в **кластеры**, а также выделение «типичного представителя» (прототипа) для каждого класса.

Тесная связь задачи *кластеризации* с задачей *классификации* данных, а через задачу классификации — с задачей *приближенного представления* зависимостей.

Типовые НС-задачи (ХХIV)

Взаимосвязи между задачами

Существуют **тесные связи** между следующими классами задач:

- классификация** данных;
- выявление закономерностей** в данных;
- кластеризация** данных.

Эти задачи связаны с различными аспектами проблемы **распознавания образов**.

Проблема распознавания образов обычно рассматривается **как часть** более широкой проблемы — **анализ сцен**, т.е. рассмотрение не изолированных образов, а их взаимодействующих совокупностей.

Пример анализа сцен — поиск и распознавание некоторого объекта в реальной среде его существования (например, средствами авиаразведки, сочетанием визуальных, инфракрасных и радиолокационных средств, найти танк, спрятанный в лесу).

Через задачу классификации — тесная связь с задачей **приближенного представления** зависимостей.

Типовые НС-задачи (ХХV)

Сжатие данных – 1

Виды кодирования данных

Длина описания данных пропорциональна:

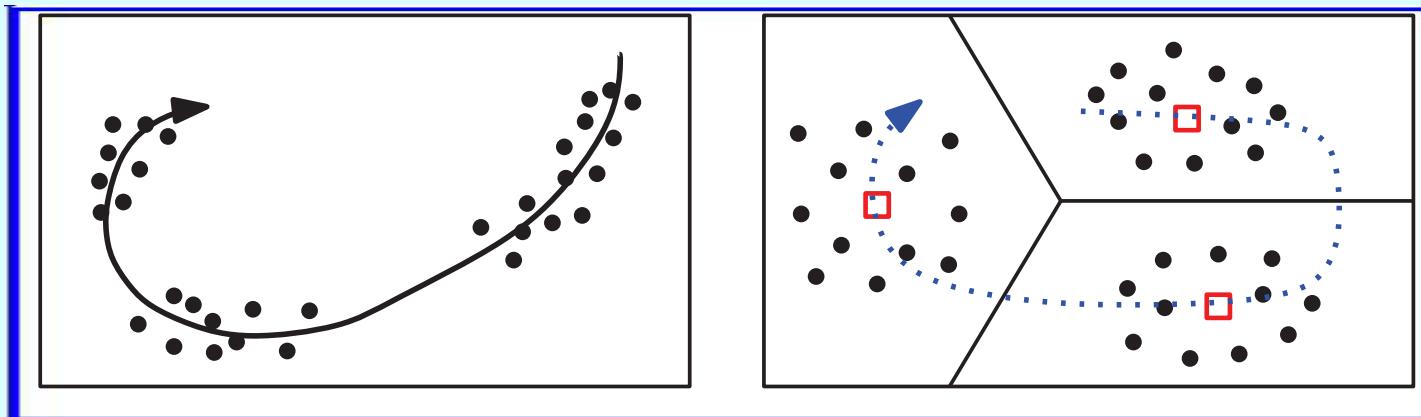
- размерности** данных d , т.е. числу компонент x_i входных векторов x ;
- разрядности** данных b (т.е. числу бит, используемых для представления компонент x_i), определяющей возможное разнообразие принимаемых ими значений.

Два предельных **вида кодирования**, использующих противоположные способы сжатия информации:

- понижение размерности** с минимальной потерей информации (например, путем выделения наборов *независимых признаков* с помощью анализа главных компонент);
- уменьшение разнообразия** данных за счет выделения конечного набора *прототипов* и отнесения данных к одному из выделенных типов (классов).

Типовые НС-задачи (XXVI)

Сжатие данных – 2



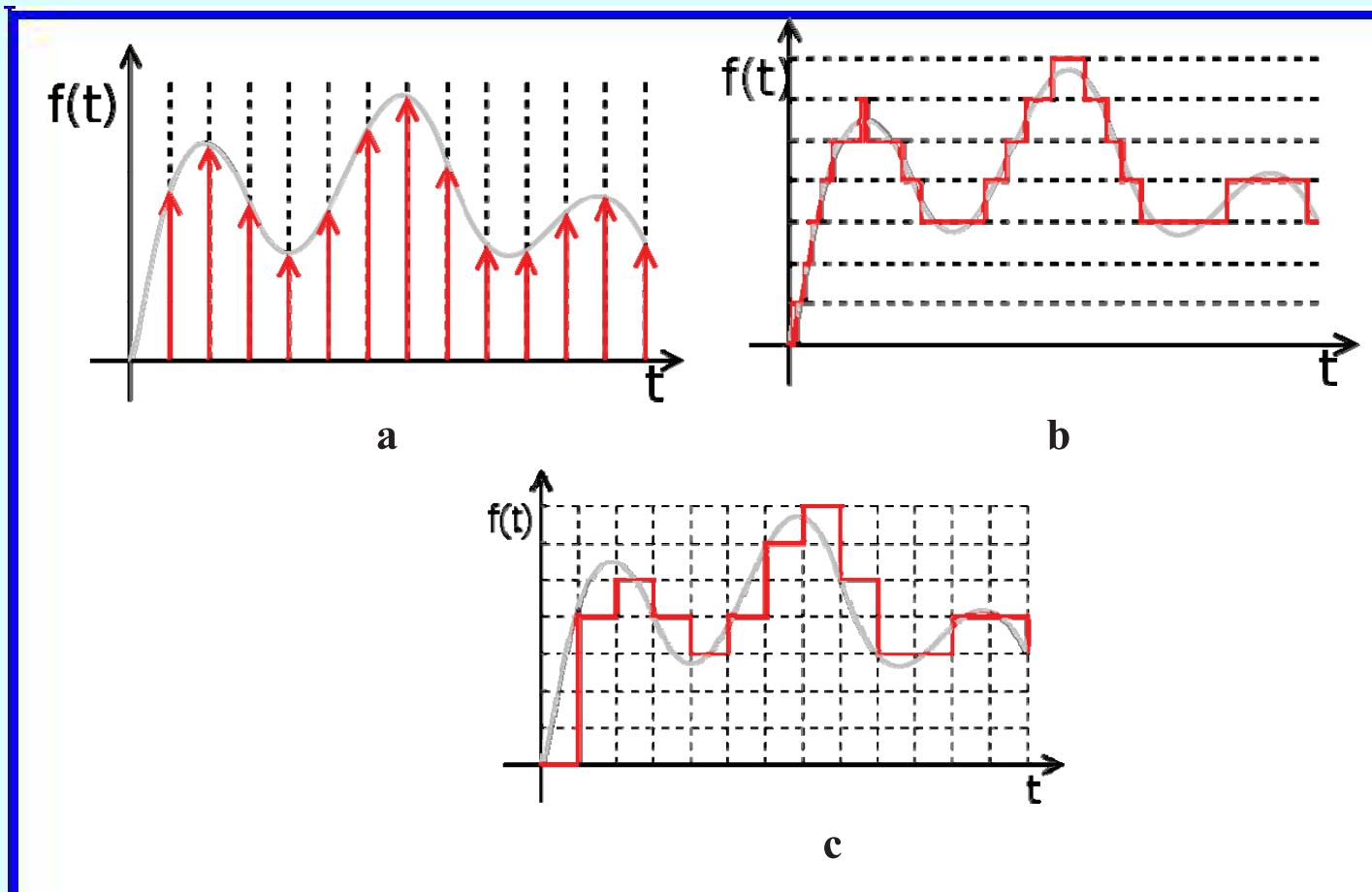
Два типа сжатия данных:

- Понижение размерности** — позволяет описывать данные меньшим числом компонент (левый рисунок).
- Квантование (кластеризация)** — позволяет снизить разнообразие данных (правый рисунок, без синего пунктира).

Комбинированный вариант — прототипы (красные квадраты на правом рисунке) упорядочены в пространстве более низкой размерности, чем исходная (синий пунктир на правом рисунке).

Типовые НС-задачи (XXVII)

Сжатие данных – 3



Дискретизация и квантование аналогового сигнала: **a** — дискретизация сигнала;
b — квантование сигнала; **c** — цифровой сигнал.

Источник: Quantization (signal processing). – From Wikipedia.

Типовые НС-задачи (ХХVIII)

Сжатие данных – 4

Роль сжатия данных:

Сжатие данных — уменьшение степени их избыточности, использующее закономерности, имеющиеся в данных.

Выделение независимых признаков при сжатии данных существенно облегчает последующую работу с данными.

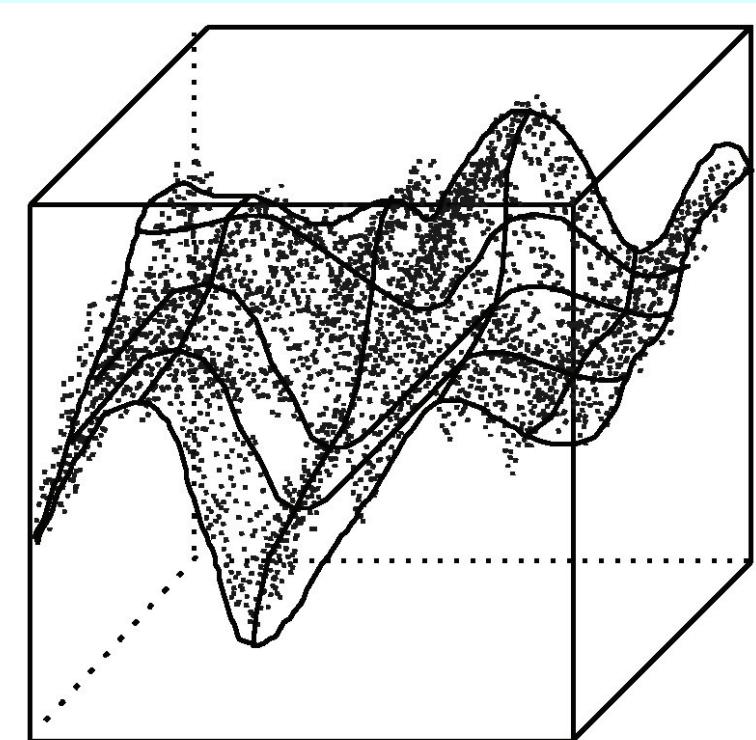
Самообучающиеся модели (в том числе и НС-модели), реализующие такое сжатие данных, чаще всего используются именно для предобработки «сырых» данных.

Такие модели имеют **адаптивный характер** и фактически кодируют входные данные наиболее компактным (при заданных ограничениях) кодом.

Типовые НС-задачи (XXIX)

Визуализация данных – 1

Двумерная топографическая карта



Двумерная топографическая карта набора многомерных данных

Упорядочение входной информации в виде **двумерной карты** (сетки).

Каждый **многомерный вектор** имеет свою координату на этой сетке.

Чем ближе координаты двух векторов на карте, тем ближе они в исходном пространстве.

Такая топографическая карта дает **наглядное представление** о структуре данных в многомерном пространстве.

На рисунке — каждая точка в **трехмерном пространстве** попадает в свою ячейку сетки, имеющую координату ближайшего к ней элемента **двумерной сетки**.

Источник: Ежов А.А., Шумский С.А. Нейрокомпьютинг и его приложения в экономике и бизнесе. – М.: Изд-во МИФИ, 1998. – 224 с. (рис.12, с.85)

Типовые НС-задачи (XXX)

Визуализация данных – 2

Одномерная топографическая карта

Одномерная топографическая карта набора двумерных данных

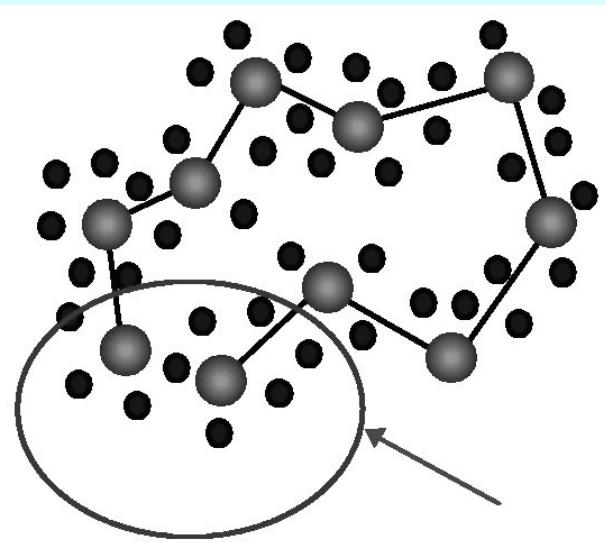
Упорядочение входной информации в виде **одномерной карты**.

Каждый **многомерный вектор** имеет свою координату на этой сетке.

Чем ближе координаты двух векторов на карте, тем ближе они в исходном пространстве, **но не наоборот**.

Стрелкой на рисунке показана **область нарушения непрерывности** отображения: **близкие на плоскости** точки отображаются на **противоположные концы карты**.

Источник: Ежов А. А., Шумский С. А. Нейрокомпьютинг и его приложения в экономике и бизнесе. – М.: Изд-во МИФИ, 1998. – 224 с. (рис.13, с.85)

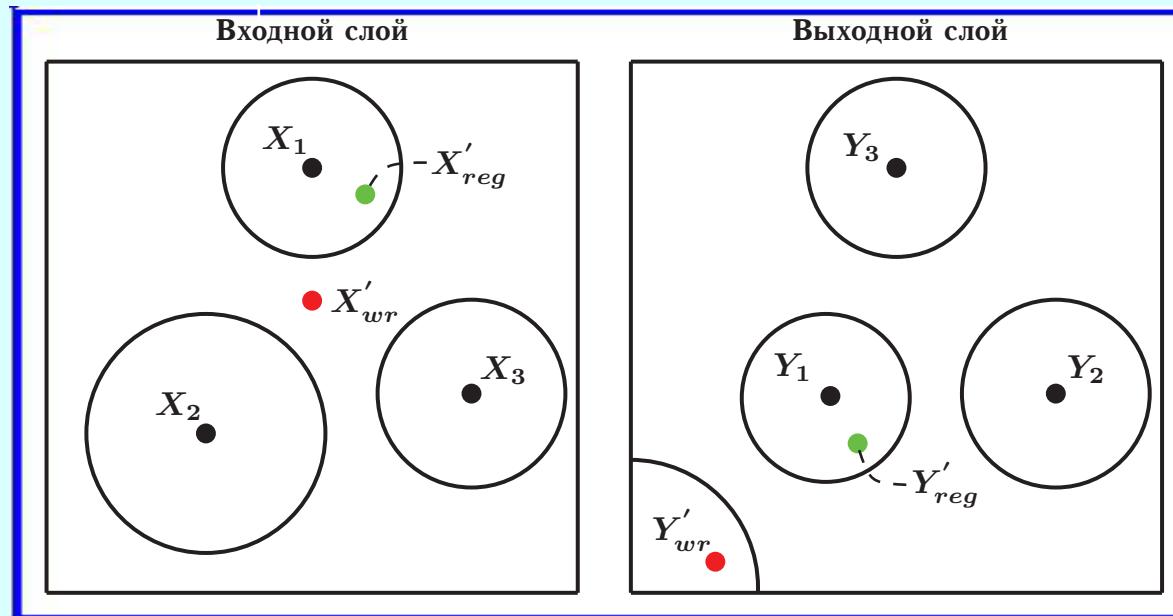


Используются **отображения многомерных данных** на одномерные, двумерные и трехмерные (третья координата — **цвет**) сетки.

Типовые НС-задачи (XXXI)

Ассоциативная память – 1

Гетероассоциативная память (1)



X_i , Y_i — эталоны (прототипы) классов;
 X'_{reg} — «свой» паттерн; X'_{wr} — «чужой» паттерн

Источник: Фролов А.А. Структура и функции обучающихся нейронных сетей // В сб.: Нейрокомпьютер как основа мыслящих ЭВМ / Под ред. А.А. Фролова и Г.И. Шульгиной. – М.: Наука, 1993. – с.92–110 (рис.3, с.103).

Типовые НС-задачи (XXXII)

Ассоциативная память – 2

Гетероассоциативная память (2)

Функции гетероассоциативной памяти:

- **В режиме записи:** для запоминания ей последовательно предъявляется L пар входных и выходных паттернов активности (X_i, Y_i). Векторы $\textcolor{red}{X}_i$ и $\textcolor{red}{Y}_i$ называются *эталонами* (прототипами, образцами, центрами классов).
- **В режиме воспроизведения:** на входе системы последовательно активируются паттерны X'_j . На основе некоторого *решающего правила* среди векторов X'_j распознаются «*свой*» $\textcolor{red}{X}'_{reg}$ (принадлежат заданной окрестности эталонов X_i) и «*чужие*» $\textcolor{red}{X}'_{wr}$ (не принадлежат ей).

Если паттерн X' **распознан** решающим правилом как «*свой*» $\textcolor{red}{X}'_{reg}$, то на выходе системы **активируется** паттерн $\textcolor{red}{Y}'_{reg}$, близкий к эталону Y_i ,льному для эталона X_i , близким к которому оказался входной паттерн X'_{reg} .

Интерпретация гетероассоциативной памяти:

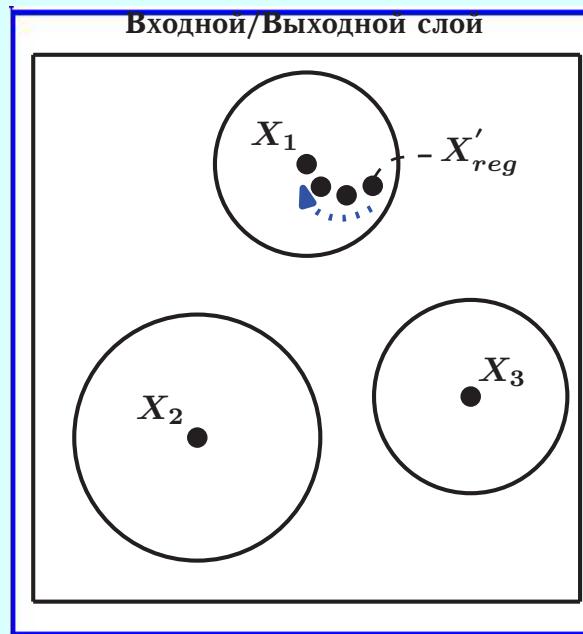
нейрофизиология — условный рефлекс (стимул → реакция)

информатика — адресация по содержимому (ключевое слово → искомые данные)

Типовые НС-задачи (XXXIII)

Ассоциативная память – 3

Автоассоциативная память (1)



X_i — эталоны (прототипы) классов

X'_{reg} — паттерн, распознаваемый как «свой»

Синий пунктир — многотактовое воспроизведение эталона X_i

Источник: [Фролов А.А.](#) Структура и функции обучающихся нейронных сетей // В сб.: Нейрокомпьютер как основа мыслящих ЭВМ / Под ред. А.А. Фролова и Г.И. Шульгиной. – М.: Наука, 1993. – с.92–110 (рис.3, с.103).

Типовые НС-задачи (XXXIV)

Ассоциативная память – 4

Автоассоциативная память (2)

Функции автоассоциативной памяти:

- В режиме записи:** для запоминания ей последовательно предъявляются L входных векторов \mathbf{X}_i (эталонов, прототипов).
- В режиме воспроизведения:** система распознает (как и гетероассоциативная память) среди паттернов, активированных на входе, «**свой**» и «**чужие**».

Если паттерн \mathbf{X}' **распознан** решающим правилом как «**свой**» \mathbf{X}'_{reg} , то на выходе системы **активируется** паттерн \mathbf{X}'' , который ближе к эталону \mathbf{X}_i , чем входной паттерн \mathbf{X}' , т.е. автоассоциативная память выполняет функцию **коррекции**.

Интерпретация автоассоциативной памяти:

коррекция — «испорченный» входной паттерн \mathbf{X}' (т.е. отличающийся в деталях от эталона) цепочкой переходов $\mathbf{X}' \rightarrow \mathbf{X}'' \rightarrow \mathbf{X}''' \rightarrow \dots$ переводится в этalon \mathbf{X}_i ;

восстановление — в качестве паттернов \mathbf{X}' можно использовать фрагменты эталонов \mathbf{X}_i , тогда *полный* эталонный паттерн будет восстановлен *по его части*.

Типовые НС-задачи (XXXV)

Оптимизация – 1

Задача целочисленного программирования

Задача математического программирования:

$$\left\{ \begin{array}{l} \min_x f(x) \\ \text{при ограничениях:} \\ g_j(x) \leq 0, \quad j = 1, 2, \dots, p, \\ h_k(x) = 0, \quad k = 1, 2, \dots, q. \end{array} \right.$$

Целочисленное программирование (ЦП) — это один из разделов математического программирования, в котором **все переменные** принимают **только целочисленные значения**.

Хотя в принципе задача ЦП может быть решена **полным перебором** всех допустимых решений, **такой путь непрактичен** для задач ЦП реальной размерности.

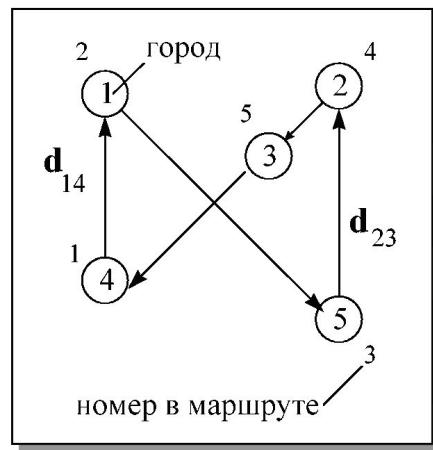
Метод ветвей и границ — задачи линейного ЦП.

Типовые НС-задачи (XXXVI)

Оптимизация – 2

Задача целочисленного программирования

Порядок прохождения городов



Нейронная кодировка маршрута

город	1	2	3	4	5
1	0	1	0	0	0
2	0	0	0	1	0
3	0	0	0	0	1
4	1	0	0	0	0
5	0	0	1	0	0

номер в маршруте

Задача коммивояжера

Найти **маршрут наименьшей длины**, проходящий через все города и ни через один из них не проходящий более одного раза.

Слева — один из возможных *маршрутов* коммивояжера в случае задачи с 5 городами.

Справа — кодирование этого маршрута состояниями 25 бинарных нейронов сети Хопфилда.

Источник: Ежов А. А., Шумский С. А. Нейрокомпьютинг и его приложения в экономике и бизнесе. – М.: Изд-во МИФИ, 1998. – 224 с. (рис.1, с.112)

Общий подход к решению — на состояниях элементов сети определяется функция, которую можно трактовать как **функцию энергии** сети. Требуется минимизировать эту функцию **при дополнительных условиях** — любой город в маршруте встречается лишь *однажды*, маршрут проходит через *каждый* город.

Типовые НС-задачи (XXXVII)

Основные модельные задачи

Основные модельные задачи:

- приближенное представление зависимостей;
- классификация/распознавание объектов.

Задачи приближенного представления зависимостей:

- определяют **взаимосвязи** между величинами, описывающими требуемый объект (систему);
- могут иметь как **статический**, так и **динамический** характер;
- играют **важнейшую роль** в проблемах науки и практики.

Подходы к обучению НС-моделей (I)

Обучение НС-моделей – 1

Алгоритм обучения — последовательность действий (процедура) для модификации синаптических весов и смещений НС-модели.

Цель алгоритма обучения — настроить НС-модель таким образом, чтобы она решала поставленную задачу.

Основные подходы к обучению НС-моделей:

- обучение с учителем** (supervised learning);
- обучение без учителя** (unsupervised learning);
- обучение с подкреплением** (reinforcement learning).

Подходы к обучению НС-моделей (II)

Обучение НС-моделей – 2

Обучение с учителем: известны векторные входы НС-модели p_j и выходы t_j , отвечающие этим входам, т.е. определено **требуемое поведение** сети в виде обучающего набора $\{\langle p_1, t_1 \rangle, \dots, \langle p_j, t_j \rangle, \dots, \langle p_q, t_q \rangle\}$. **Цель обучения** — так подобрать значения весов и смещений сети, чтобы выходы сети были максимально близки заданным в обучающем наборе t_j . **Типичная задача** — приближенное представление зависимостей.

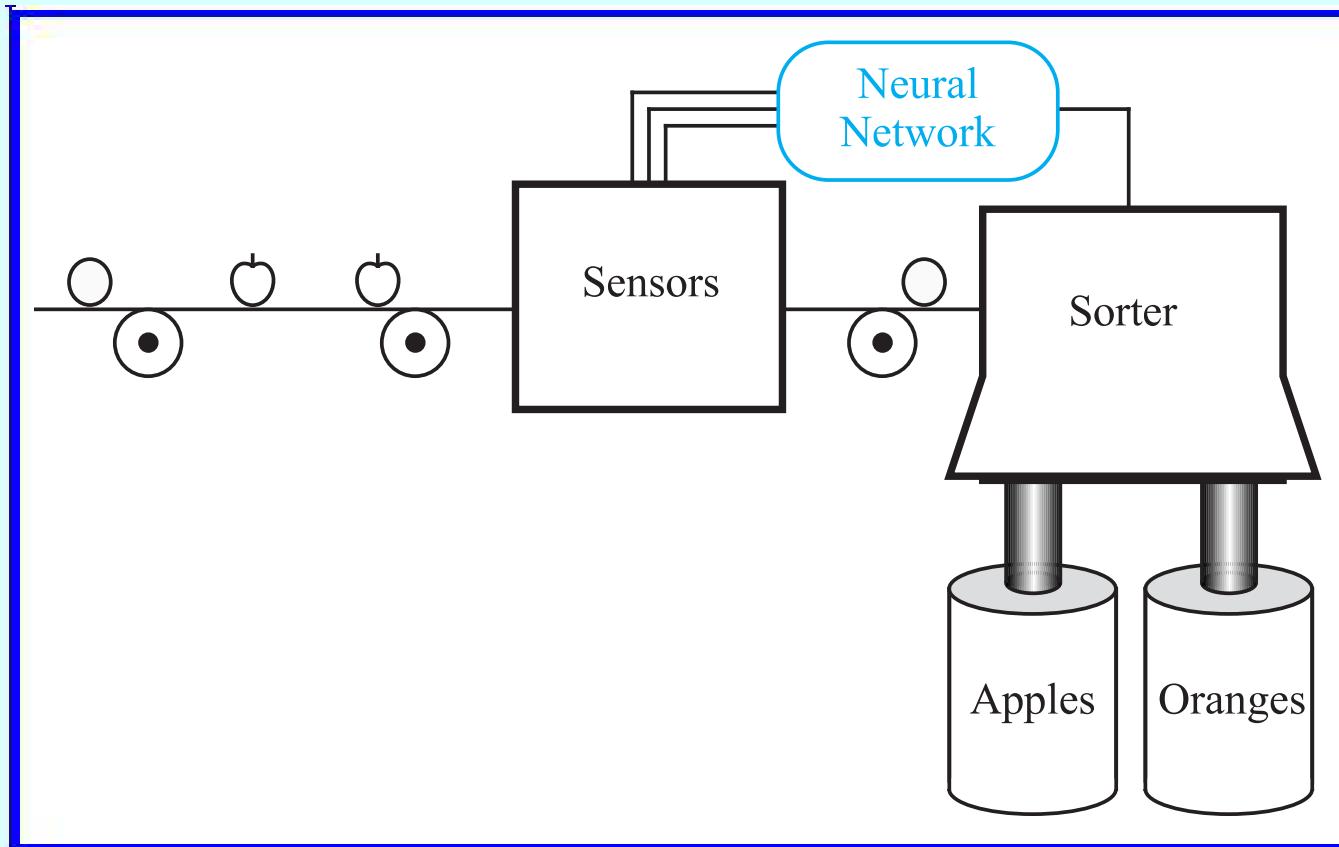
Обучение без учителя: известны **только** векторные входы НС-модели p_j , значения выходов t_j , отвечающих этим входам, отсутствуют. В этом случае обучающий набор принимает вид $\{p_1, \dots, p_j, \dots, p_q\}$. **Цель обучения** — сгруппировать элементы, входящие в обучающий набор, основываясь на закономерностях, присущих этому набору. **Типичная задача** — классификация (категоризация) данных.

Обучение с подкреплением: по общей идее сходно с обучением с учителем, но вместо точных значений выходов t_j , отвечающих соответствующим входам p_j , даются некоторые **оценки** того, насколько «хорошо» формируемая модель ведет себя на некоторой **последовательности входов**, воспринимаемых НС-моделью. **Типичная задача** — управление поведением динамических систем в среде с неопределенностями.

Иллюстративный пример (I)

Постановка задачи – 1

Задача классификации – сортировка фруктов



Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, pp.3-2–3-3).

Иллюстративный пример (II)

Постановка задачи – 2

Элементы задачи классификации

Два класса объектов — яблоки и апельсины.

Свойства объекта (определяются сенсорами):

- форма** (shape) — шарообразная (+1), эллипсоидальная (-1);
- текстура** (texture) — гладкая (+1), шероховатая (-1);
- вес** (weight) — больше одного фунта (+1), меньше одного фунта (-1).

Входной вектор нейросети:

$$\begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix} \quad \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Слева направо: вектор признаков, эталон (прототип) для класса «апельсины», эталон (прототип) для класса «яблоки».

Задача: сенсоры измеряют значения компонент вектора признаков для подаваемых фруктов, построить **нейросетевой классификатор**, сортирующий эти фрукты.

Иллюстративный пример (III)

Варианты решения задачи классификации

Нейросетевые классификаторы:

- НС-классификатор на основе **персептрана**;
- НС-классификатор на основе **сети Хемминга**;
- НС-классификатор на основе **сети Хопфилда**.

Персептрон — нейронная сеть прямого распространения.

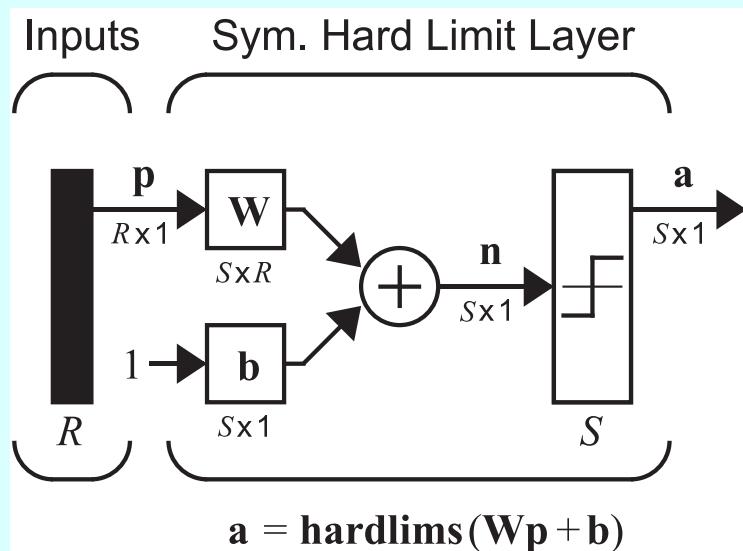
Сеть Хемминга — соревновательная нейронная сеть.

Сеть Хопфилда — рекуррентная сеть.

Иллюстративный пример (IV)

Персептрон – 1

Однослоиный персептрон общего вида



Симметричная пороговая функция:

$$\text{hardlims} = \begin{cases} a = -1, & n < 0; \\ a = +1, & n \geq 0. \end{cases}$$

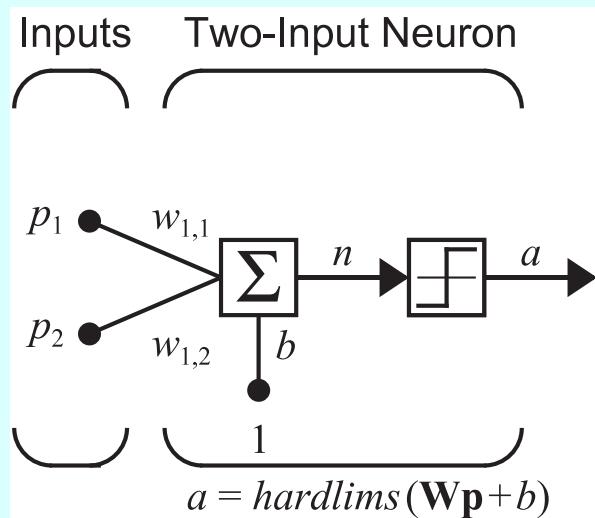
$$p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix}$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Fig.3.1, p.3-3).

Иллюстративный пример (V)

Персептрон – 2

Персептрон с двумя входами



Персептрон, состоящий из **единственного нейрона**, разбивает входные векторы на **два класса**.

Например, если **веса связей**

$$w_{1,1} = -1 \text{ и } w_{1,2} = -1$$

тогда

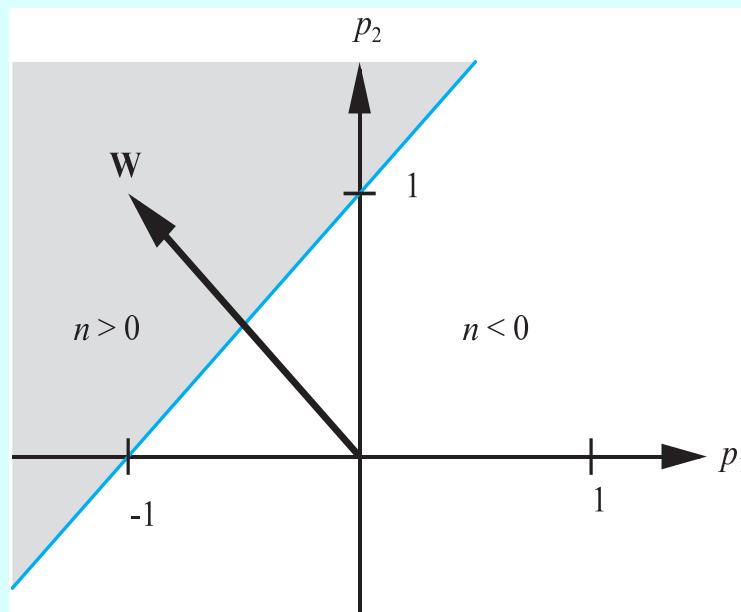
$$a = \text{hardlims}(n) = \text{hardlims}([-1 \ 1]\mathbf{p} + b).$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Fig.3.2, p.3-4).

Иллюстративный пример (VI)

Персептрон – 3

Разделяющая поверхность персептрана



Входной объект (вектор), персептрана с двумя входами:

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$$

Уравнение поверхности (линии для персептрана с двумя входами), **разделяющей классы** входных объектов (векторов):

$$n = [-1 \ 1]\mathbf{p} - 1 = 0.$$

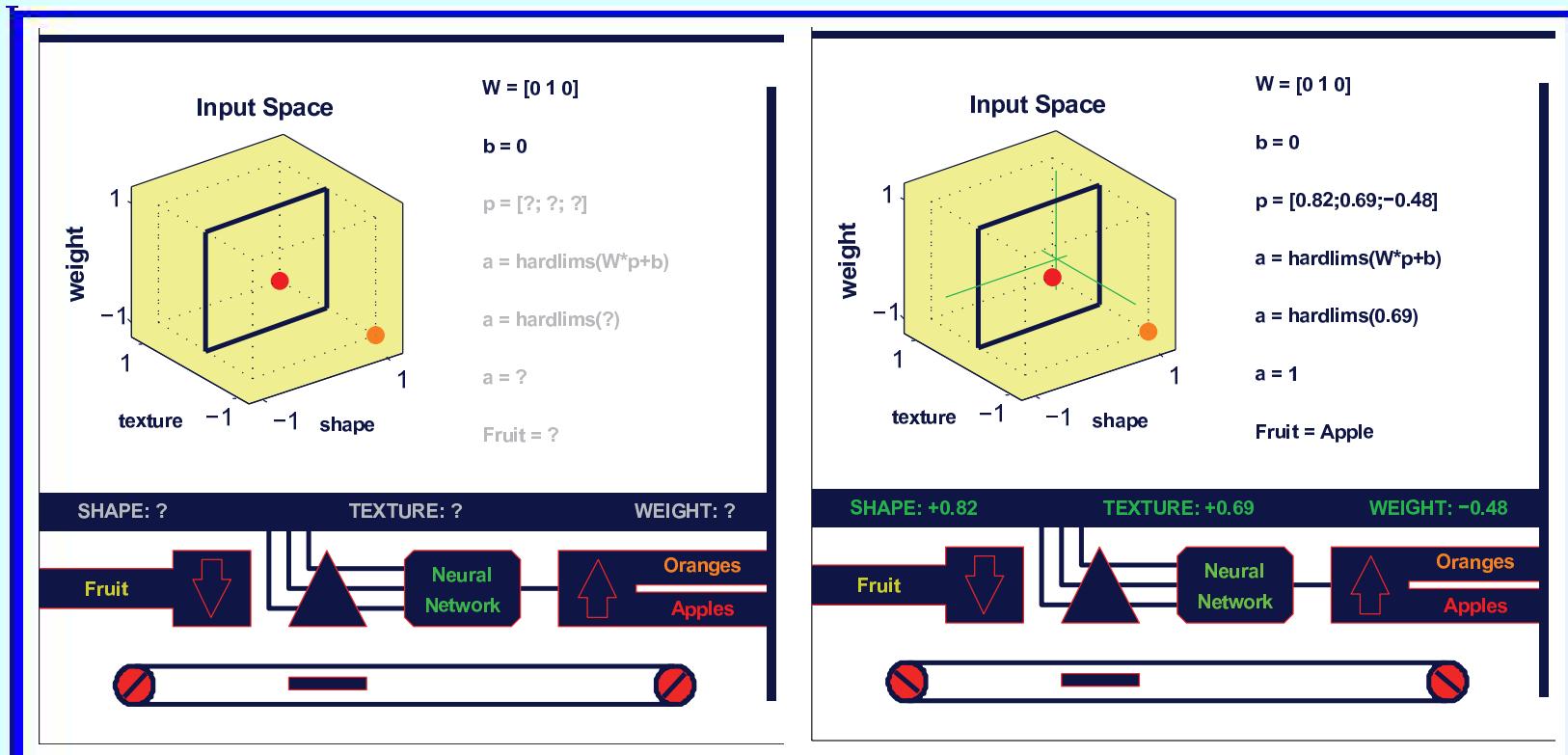
В общем случае уравнение **разделяющей поверхности** (гиперплоскости) для персептрана имеет вид:

$$\mathbf{W}\mathbf{p} + b = 0.$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Fig.3.3, p.3-5).

Иллюстративный пример (VII)

Персептрон – 4

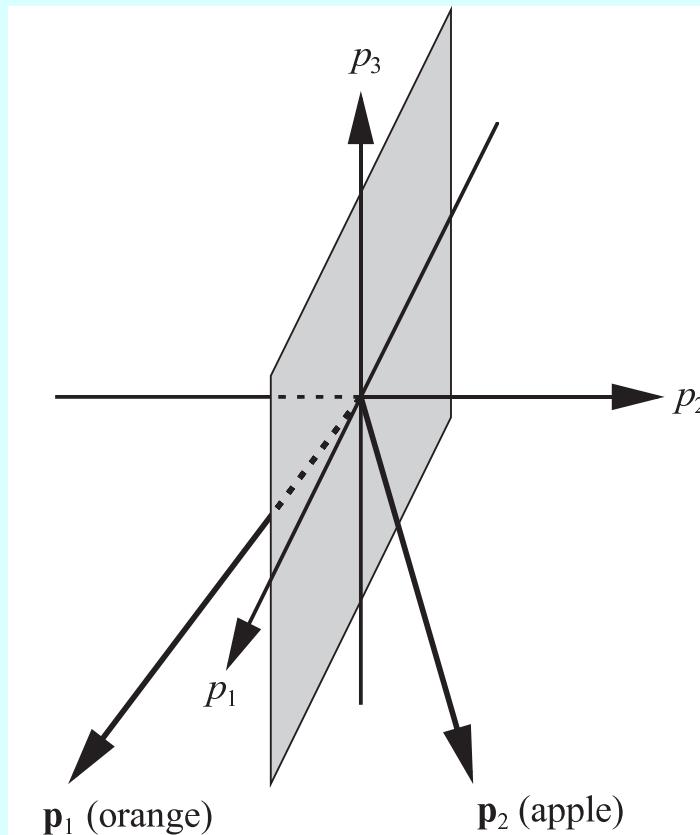


Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, pp.3-3-3-8).

Иллюстративный пример (VIII)

Персептрон – 5

Персептрон с тремя входами (1)



Уравнение персептрана с тремя входами:

$$a = \text{hardlims} \left([w_{1,1} \ w_{1,2} \ w_{1,3}] \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b \right).$$

Эталонные векторы (прототипы) классов – «апельсины» (p_1) и «яблоки» (p_2):

$$p_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \quad p_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

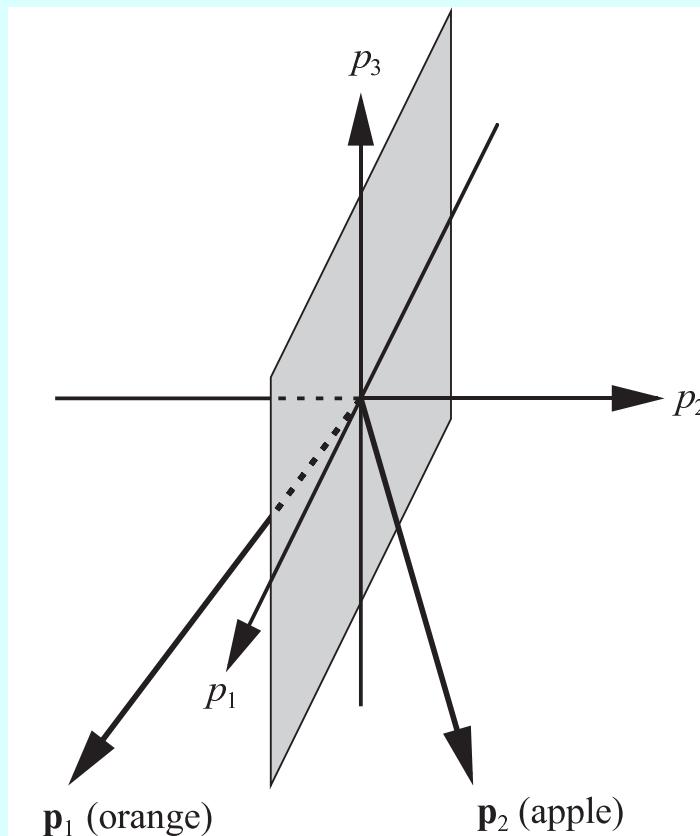
Разделяющая поверхность – плоскость $p_2 = 0$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Fig.3.4, p.3-6).

Иллюстративный пример (IX)

Персепtron – 6

Персепtron с тремя входами (2)



Разделяющая поверхность — плоскость p_1, p_2 , т.е.

$$p_2 = 0$$

или

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0.$$

Следовательно, **весовая матрица W и смещение b** определяются как

$$W = [0 \ 1 \ 0], \quad b = 0.$$

Иллюстративный пример (X)

Персепtron – 7

Персепtron с тремя входами (3)

Проверка работоспособности

Апельсин:

$$a = \text{hardlims} \left([0 \ 1 \ 0] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = \text{hardlims}(-1) = -1 \text{ (апельсин)}$$

Яблоко:

$$a = \text{hardlims} \left([0 \ 1 \ 0] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = \text{hardlims}(1) = +1 \text{ (яблоко)}$$

Иллюстративный пример (XI)

Персепtron – 8

Персепtron с тремя входами (4)

Проверка работоспособности — случай «неидеального» входного вектора

Входной вектор:

$$p = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

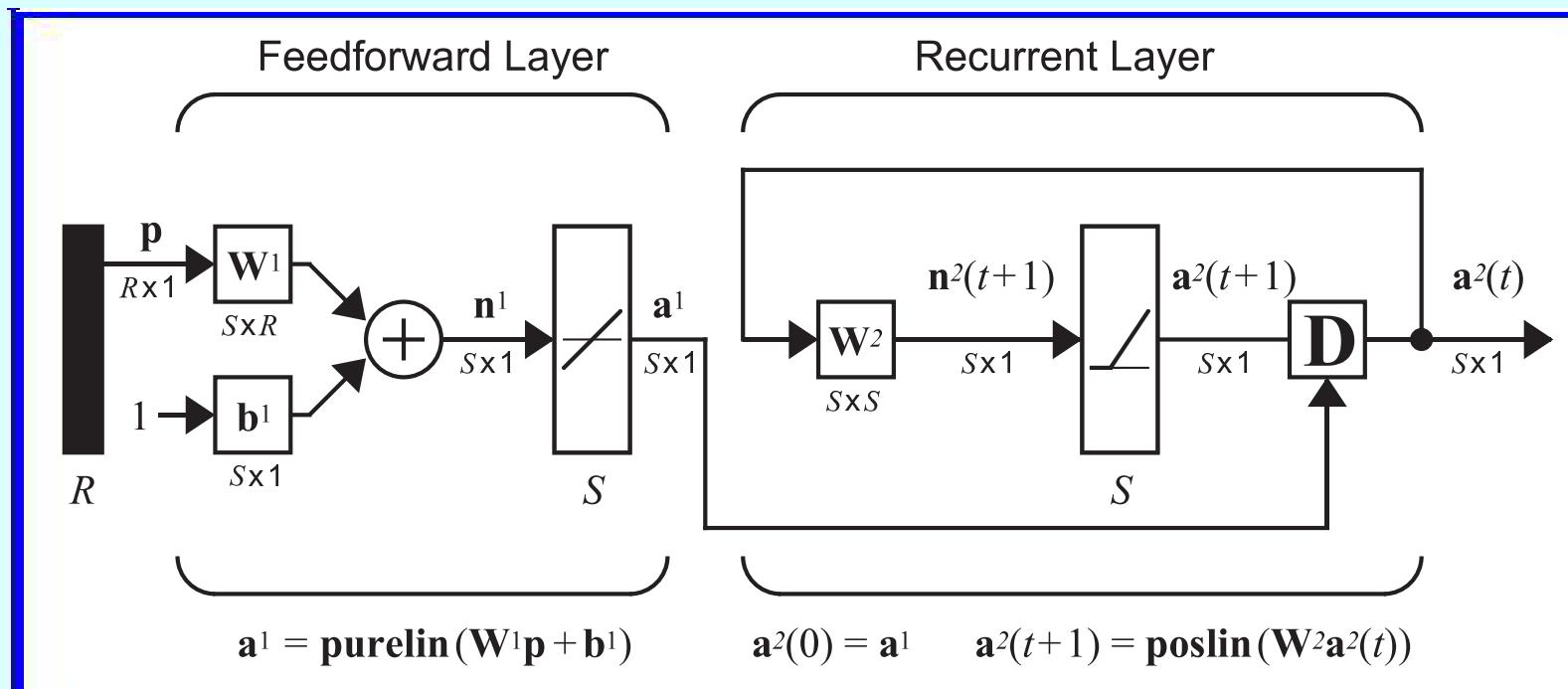
Отклик сети:

$$a = \text{hardlims}\left([0 \ 1 \ 0] \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = \text{hardlims}(-1) = -1 \text{ (апельсин)}$$

p_1 — форма объекта: шарообразная (+1) или эллипсоидная (-1)

Иллюстративный пример (XII)

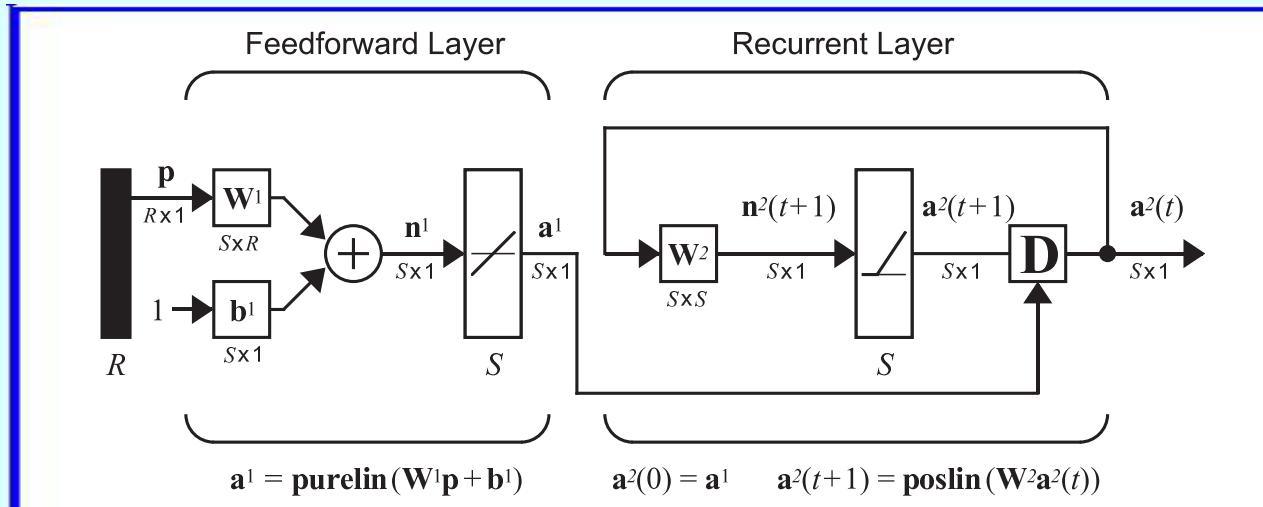
Сеть Хемминга – 1



Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Fig.3.5, p.3-8).

Иллюстративный пример (XIII)

Сеть Хемминга – 2



Сеть Хемминга — комбинация из двух подсетей (прямого распространения и рекуррентной), она решает задачу **классификации паттернов**, представленных соответствующими входными векторами.

Каждый из классов представлен своим **вектором-прототипом** (эталоном).

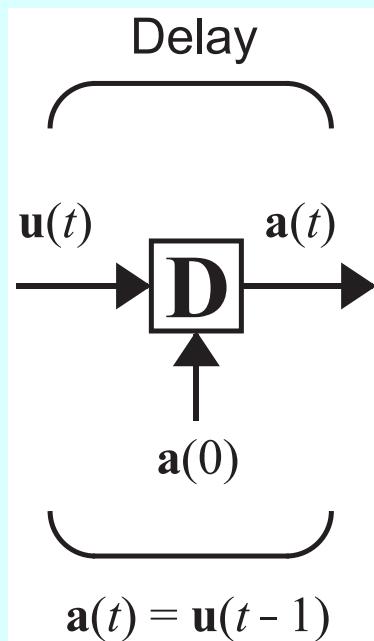
Сеть Хемминга определяет, какой из векторов-прототипов **ближе всего** к предъявленному входному вектору.

Число **S** нейронов в обоих слоях сети **совпадает** и равно числу векторов-прототипов.

Решение: выход нейрона рекуррентного слоя — после того, как процесс в рекуррентном слое завершился, остается **только один нейрон** с ненулевым выходом.

Иллюстративный пример (XIV)

Сеть Хемминга – 3



Элемент задержки

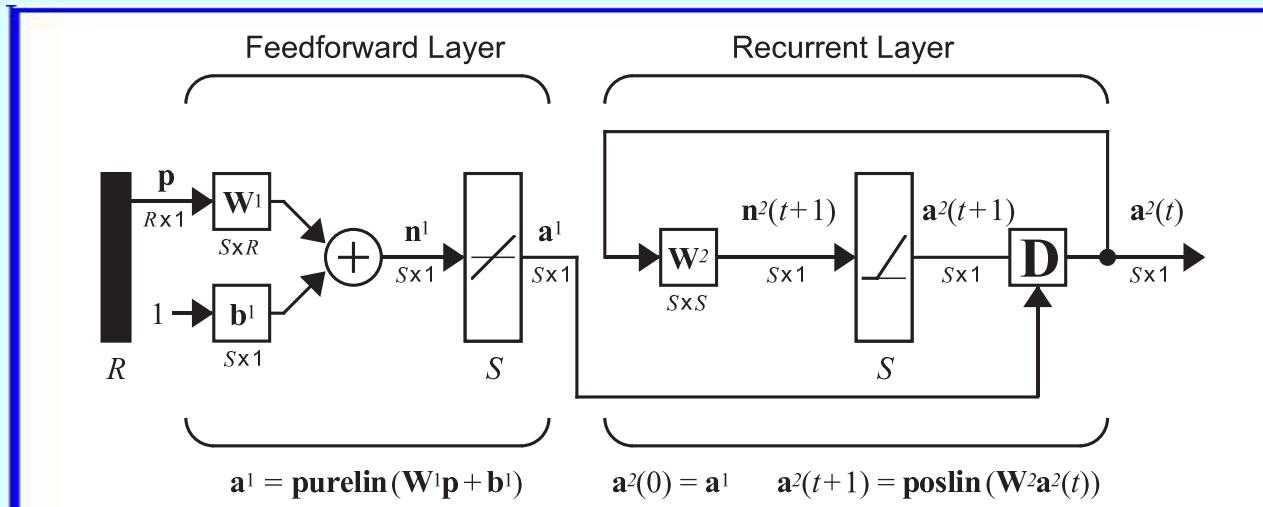
(выход элемента задержки — это его вход, задержанный на **один временной шаг**)

$$a(t) = u(t - 1)$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 2, Fig.2.11, p.2-13).

Иллюстративный пример (XV)

Сеть Хемминга – 4

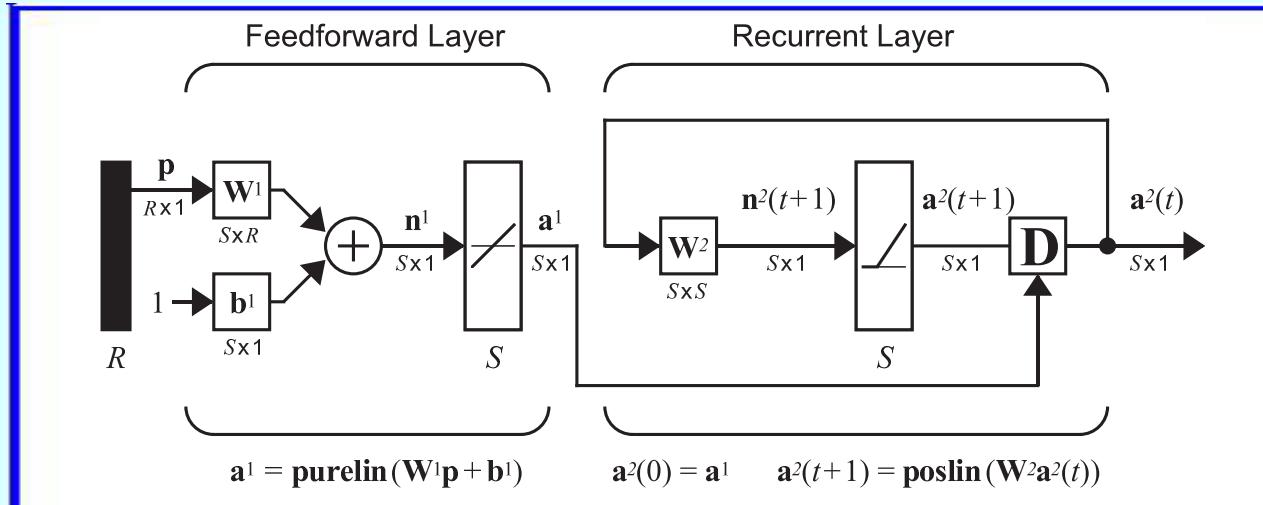


Активационная функция рекуррентного слоя
(положительная линейная функция)

$$\text{poslin} = \begin{cases} a = 0, & n < 0; \\ a = n, & n \geq 0. \end{cases}$$

Иллюстративный пример (XVI)

Сеть Хемминга – 5



Слой прямого распространения

Векторы-прототипы (эталоны классов) — строки весовой матрицы \mathbf{W}^1

$\mathbf{p}_1 = (1 \ -1 \ -1)^T$ — этalon апельсина, $\mathbf{p}_2 = (1 \ 1 \ -1)^T$ — этalon яблока

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}, \quad \mathbf{b}^1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$b_1^1 = b_2^1 = R = 3$ — из условия неотрицательности выходов слоя

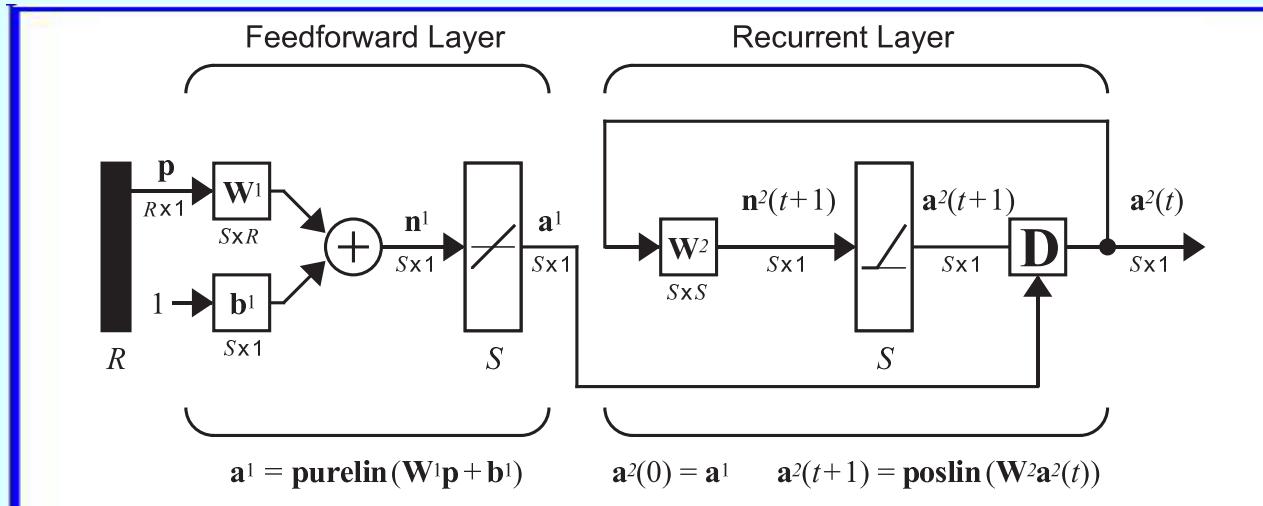
Выход слоя прямого распространения:

$$a^1 = \mathbf{W}^1 p + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} p + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T p + 3 \\ \mathbf{p}_2^T p + 3 \end{bmatrix}$$

Слой вычисляет **скалярные произведения** $a_i = (\mathbf{p}_i, \mathbf{p})$, $i = 1, 2$
прототипов $\mathbf{p}_1, \mathbf{p}_2$ и входа \mathbf{p} .

Иллюстративный пример (XVII)

Сеть Хемминга – 6



Слой прямого распространения

Слой прямого распространения вычисляет **скалярные произведения**
 $a_i = (\mathbf{p}_i, \mathbf{p})$, $i = 1, 2$ векторов-прототипов \mathbf{p}_1 и \mathbf{p}_2 и предъявляемого для классификации входного вектора \mathbf{p} .

Это **степени рассогласования** входного вектора с каждым из эталонов.

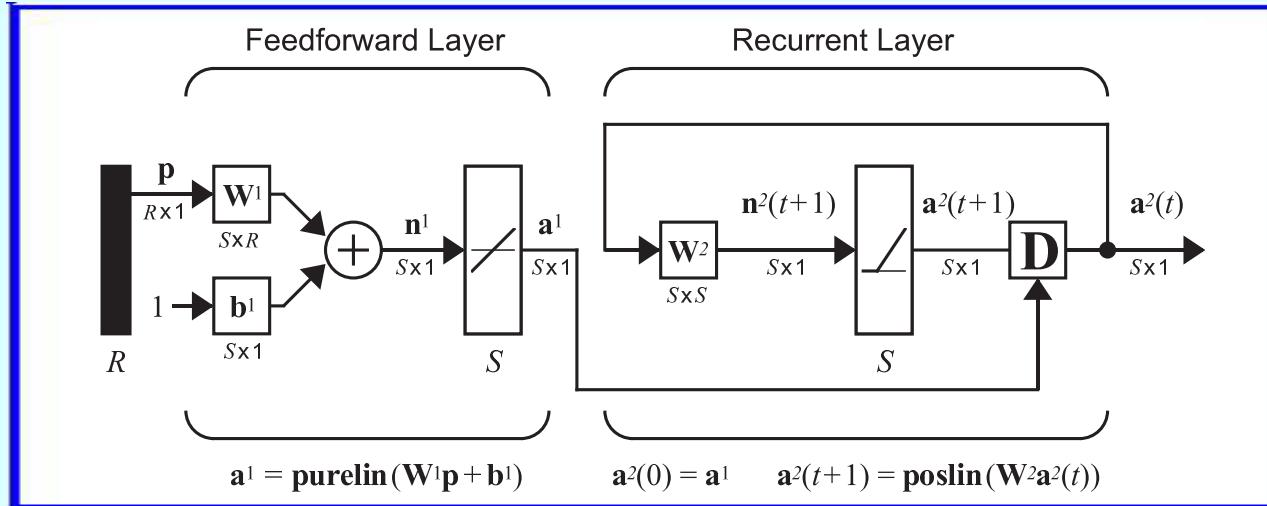
$$(\mathbf{p}_i, \mathbf{p}) = |\mathbf{p}_i| |\mathbf{p}| \cos \varphi$$

Два вектора с одинаковой длиной (нормой) имеют **наибольшее** скалярное произведение, когда они ориентированы в одном и том же направлении ($\varphi = 0$); **наименьшее** — в противоположных направлениях ($\varphi = \pi$).

Сеть Хемминга — нейрон слоя прямого распространения с наибольшим выходом соответствует паттерну-прототипу, **наиболее близкому** к входному вектору (паттерну) **в смысле расстояния Хемминга**.

Иллюстративный пример (XVIII)

Сеть Хемминга – 7



Рекуррентный слой

(реализует одну из разновидностей **соревновательного выбора**)

$$a^2(0) = a^1$$

$$a^2(t+1) = \text{poslin}(W^2 a^2(t))$$

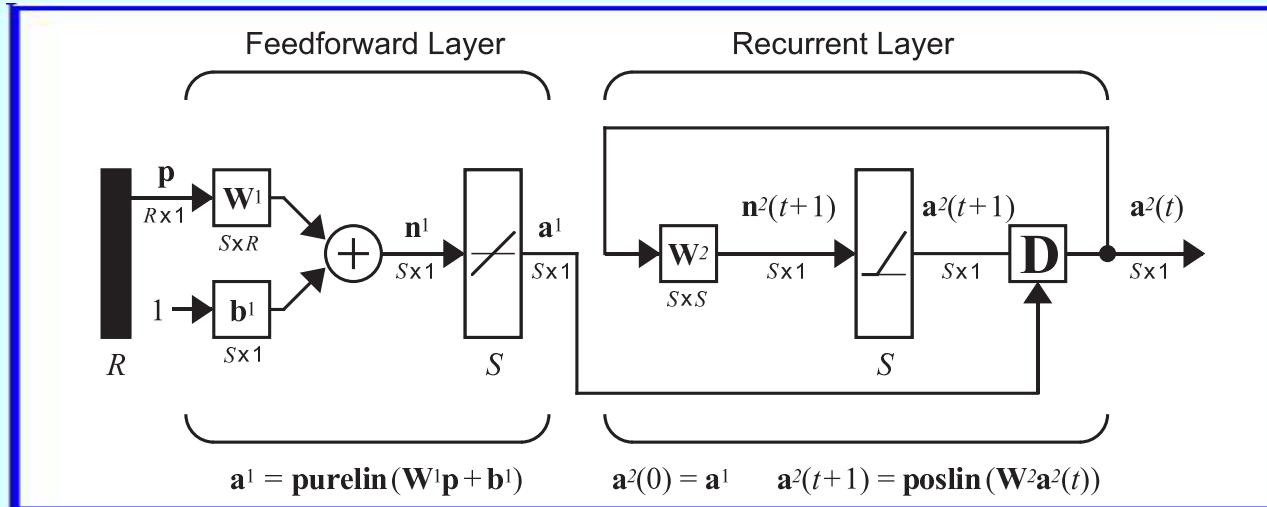
Весовая матрица рекуррентного слоя:

$$W^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}$$

Здесь $\varepsilon < 1/(S-1)$, S – число нейронов в рекуррентном слое.

Иллюстративный пример (XIX)

Сеть Хемминга – 8



Рекуррентный слой

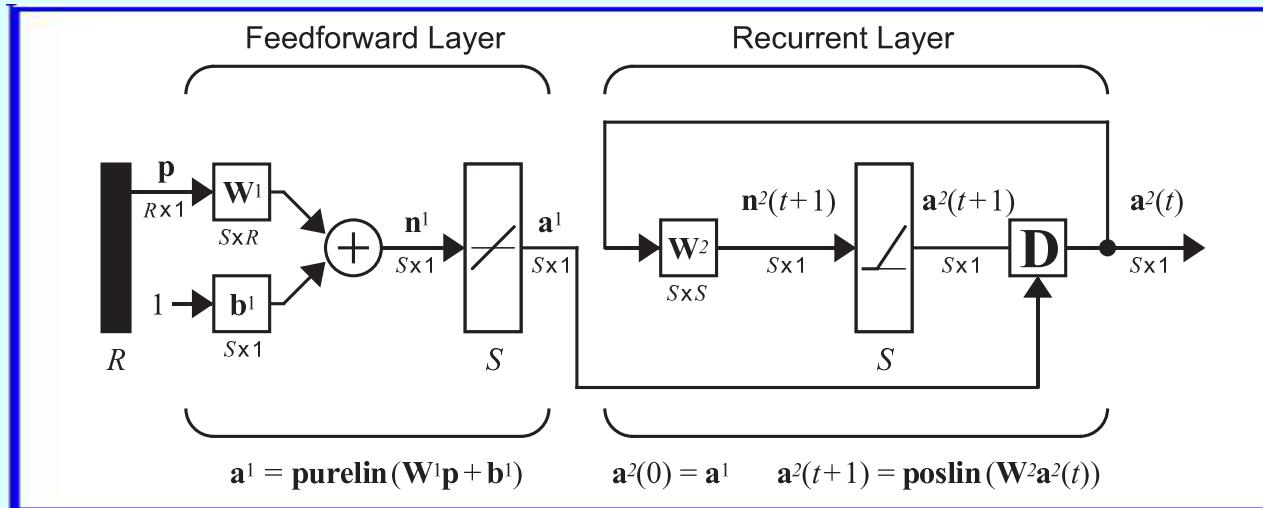
Итерация рекуррентного слоя:

$$a^2(t+1) = \text{poslin}\left(\begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix} a^2(t)\right) = \text{poslin}\left(\begin{bmatrix} a_1^2(t) - \varepsilon a_2^2(t) \\ a_2^2(t) - \varepsilon a_1^2(t) \end{bmatrix}\right)$$

С ростом номера итерации t разница между наибольшим и остальными элементами $a_i^2(t)$ **будет расти** (на каждой итерации относительные изменения постоянны, абсолютные тем больше, чем меньше величина $a_i^2(t)$). Это значит, что рекуррентный слой стремится «**обнулить**» все выходы, кроме наибольшего («**победителя**»).

Иллюстративный пример (ХХ)

Сеть Хемминга – 9



Пример работы сети Хемминга
 (задача об апельсинах и яблоках)

На входе – удлиненный апельсин:

$$p = [-1 \ -1 \ -1]^T$$

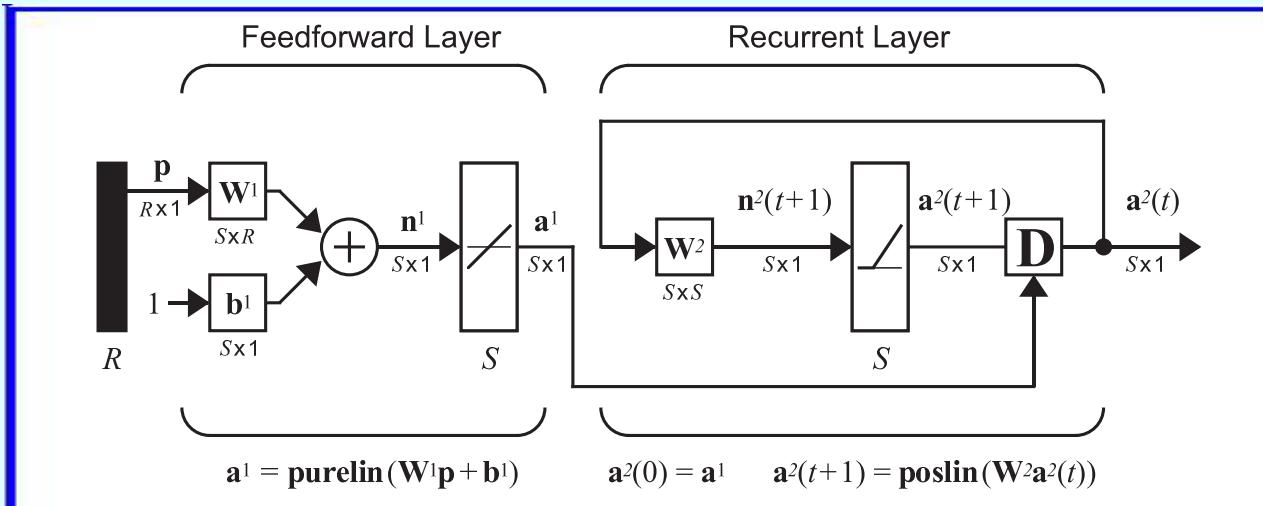
Выход слоя прямого распространения:

$$a^1 = W^1 p + b^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \\ -1 \end{bmatrix} = \begin{bmatrix} (1+3) \\ (-1+3) \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

Выходы слоя прямого распространения являются **начальными данными** для рекуррентного слоя.

Иллюстративный пример (XXI)

Сеть Хемминга – 10



Пример работы сети Хемминга

Итерации рекуррентного слоя ($\varepsilon = 0.5$):

$$a^2(1) = \text{poslin}(W^2 a^2(0)) = \text{poslin} \left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} \right) = \text{poslin} \left(\begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

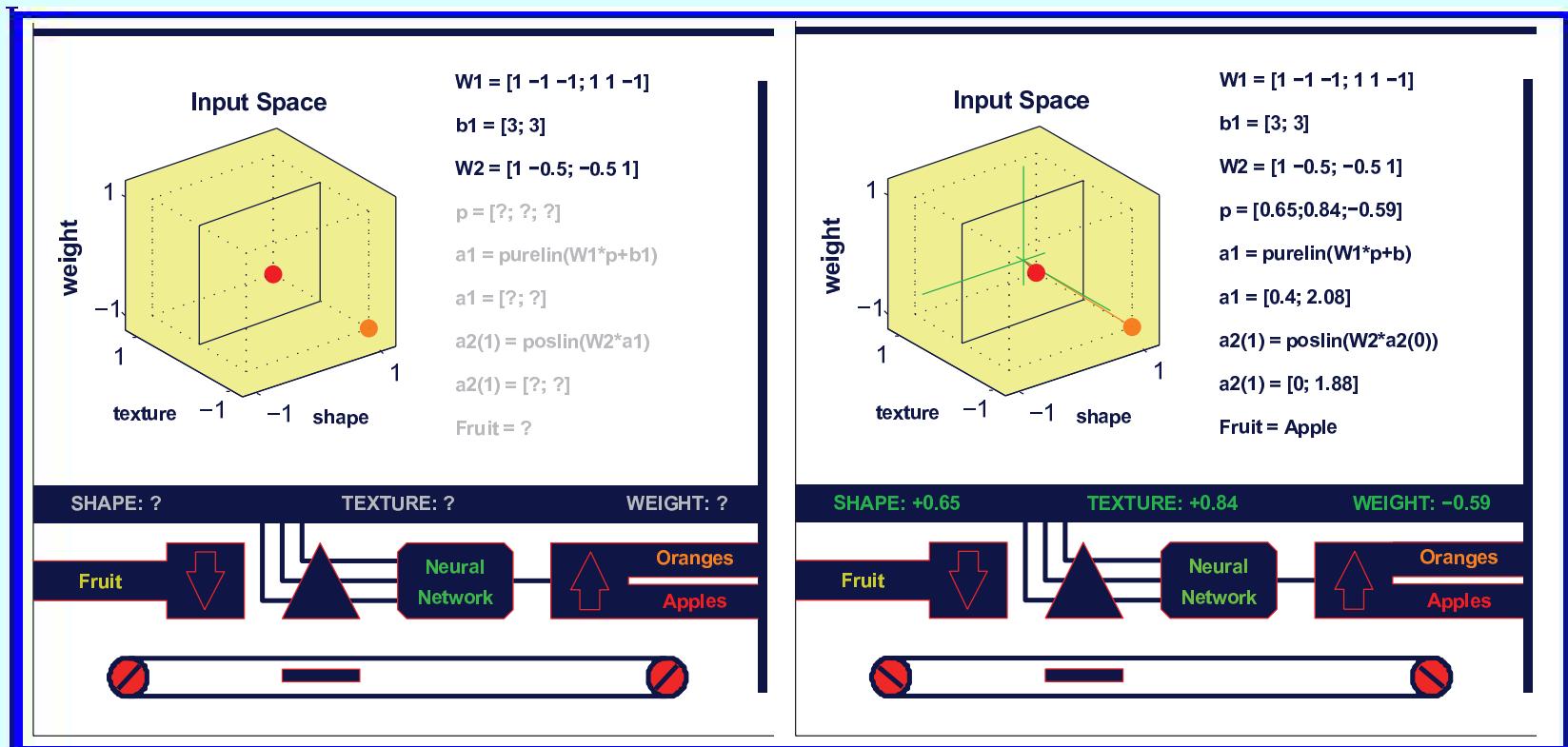
$$a^2(2) = \text{poslin}(W^2 a^2(1)) = \text{poslin} \left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) = \text{poslin} \left(\begin{bmatrix} 3 \\ -1.5 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

Значения выходов слоя остались **постоянными** при переходе от $t = 1$ к $t = 2$ — процесс сошелся.

Ненулевой выход у первого нейрона — ответ будет **«апельсин»**
 (расстояния Хемминга $d_H(p_1, p) = 1$; $d_H(p_1, p) = 2$).

Иллюстративный пример (XXII)

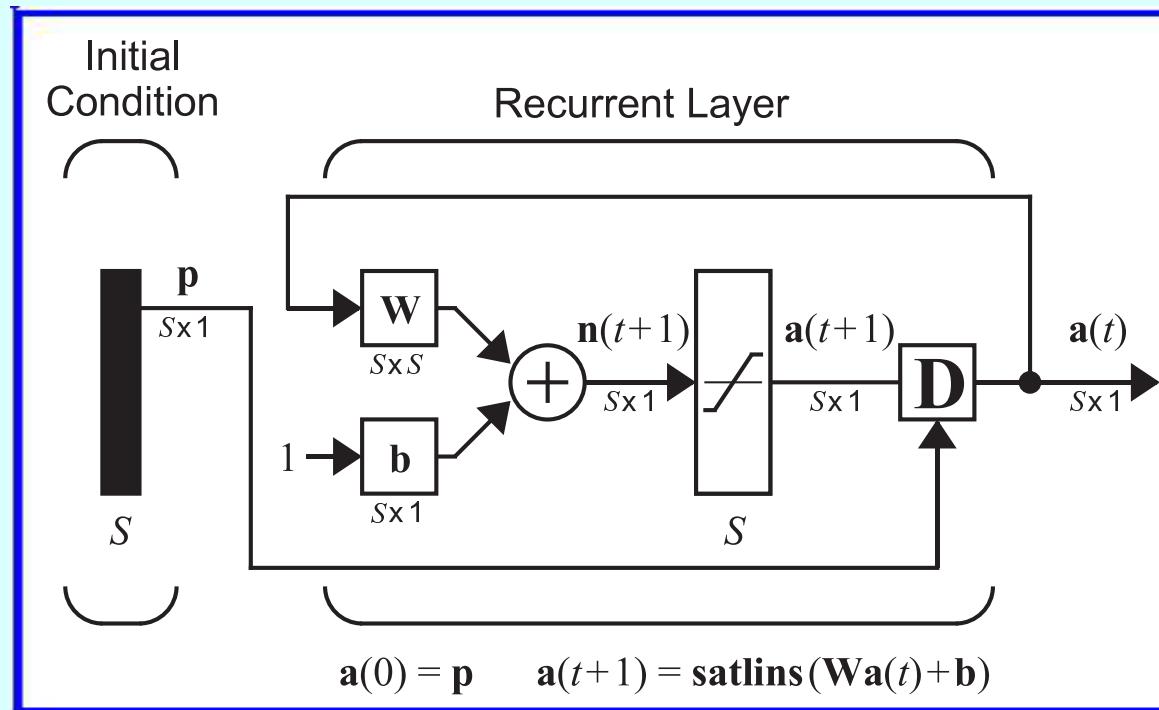
Сеть Хемминга – 11



Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, pp.3-8–3-11).

Иллюстративный пример (ХХIII)

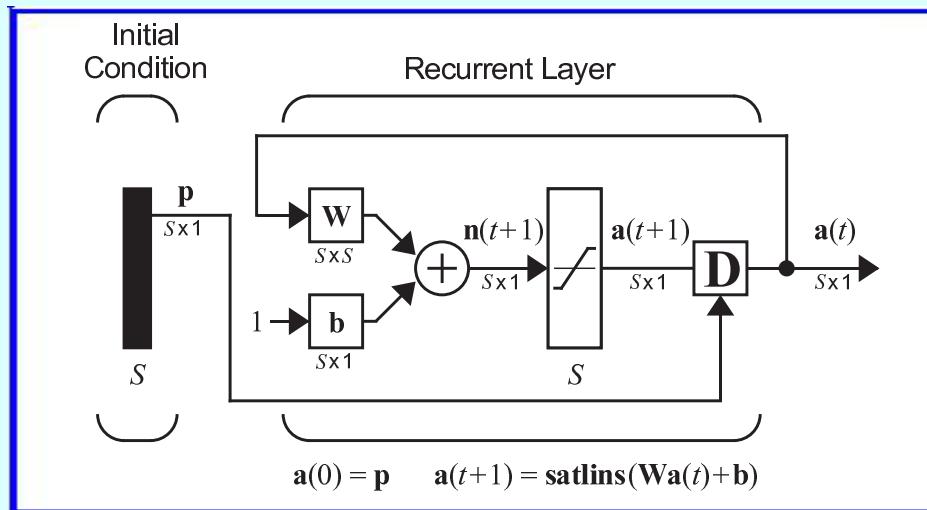
Сеть Хопфилда – 1



Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Figure 3.6, p.3-12).

Иллюстративный пример (XXIV)

Сеть Хопфилда – 2

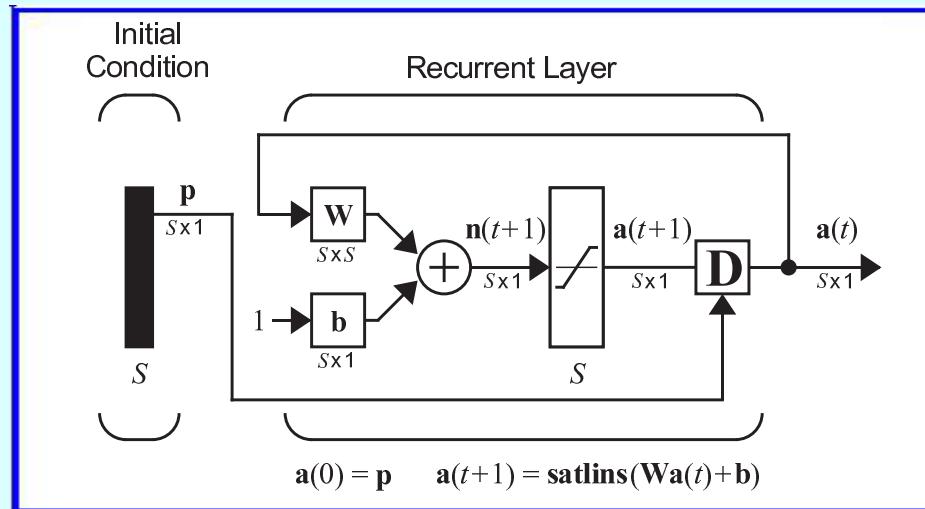


Активационная функция сети Хопфилда
(симметричная линейная функция с насыщением)

$$\text{satlins} = \begin{cases} a = -1, & n < -1; \\ a = n, & -1 \leq n \leq 1; \\ a = 1, & n > 1. \end{cases}$$

Иллюстративный пример (XXV)

Сеть Хопфилда – 3



Соотношения, описывающие работу сети:

$$a(0) = p$$

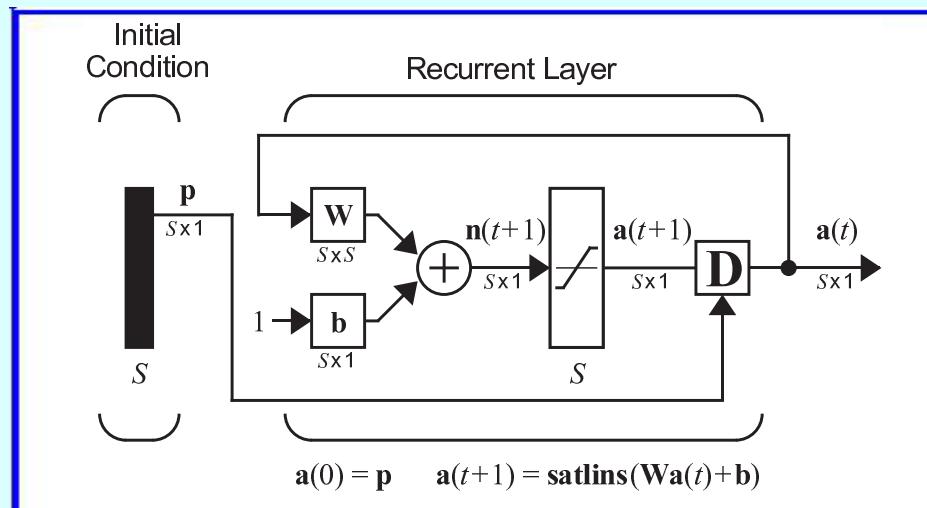
$$a(t + 1) = \text{satlins}(Wa(t) + b)$$

Весовая матрица и вектор смещения для задачи сортировки яблок и апельсинов:

$$W = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 1.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \quad b = \begin{bmatrix} 0.9 \\ 0 \\ -0.9 \end{bmatrix}$$

Иллюстративный пример (XXVI)

Сеть Хопфилда – 4



Соотношения, описывающие работу сети:

$$a(0) = p$$

$$a(t + 1) = \text{satlins}(Wa(t) + b)$$

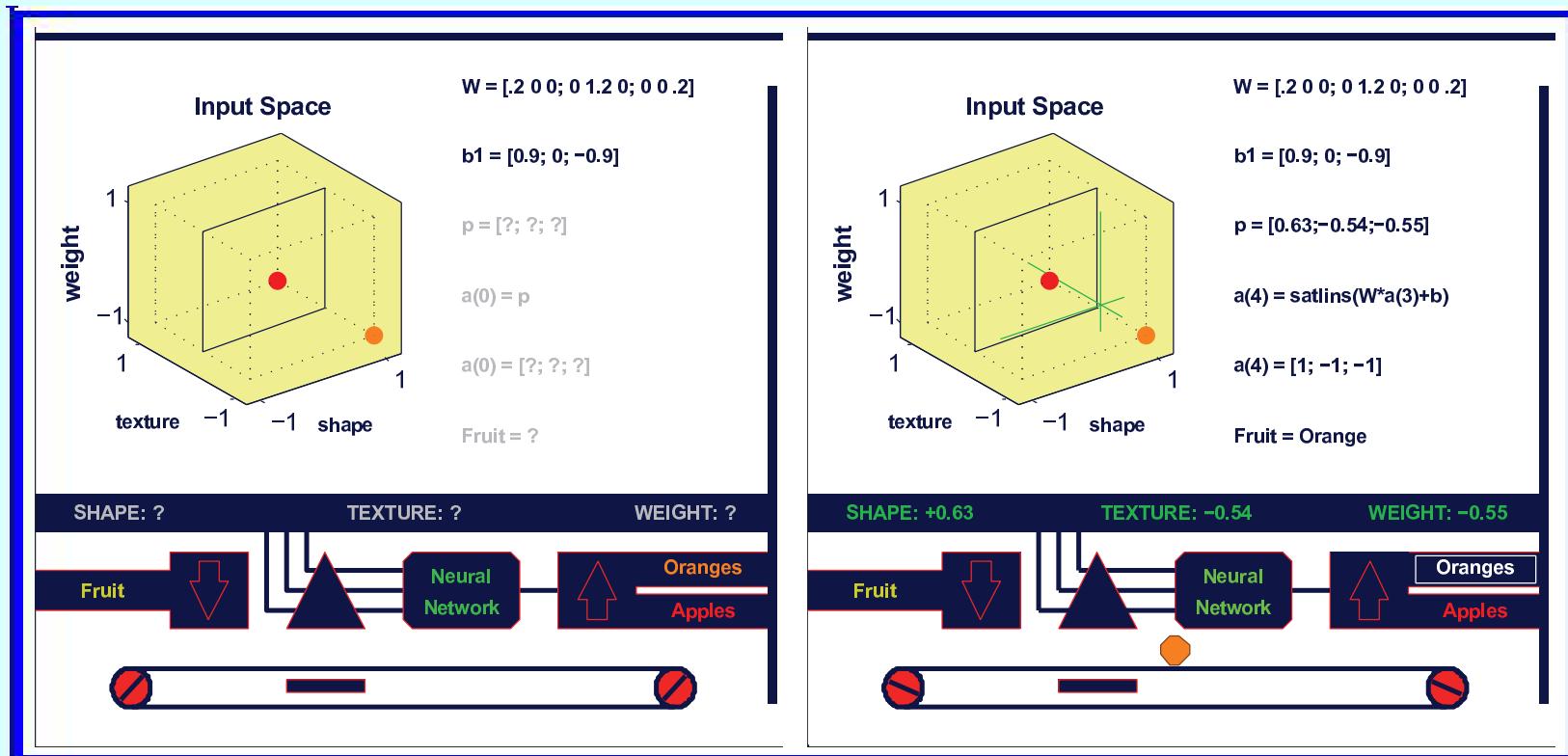
Итерации для задачи сортировки яблок и апельсинов
(входной вектор для удлиненного апельсина)

$$a(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \quad a(1) = \begin{bmatrix} 0.7 \\ -1 \\ -1 \end{bmatrix}, \quad a(2) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \quad a(3) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

Ответ: предъявленный входной вектор классифицируется как «апельсин».

Иллюстративный пример (XXVII)

Сеть Хопфилда – 5



Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, pp.3-12–3-14).

Иллюстративный пример (XXVIII)

Вид результатов, выдаваемых сетями
(в задаче сортировки яблок и апельсинов)

Персепtron Розенблатта

Сеть имеет **единственный выход**.

Он принимает значения (-1) для апельсина и $(+1)$ для яблока.

Сеть Хемминга

Число нейронов в выходном слое равно **числу распознаваемых классов**.

Ненулевой выход — только у одного выходного нейрона.

Если это первый нейрон — апельсин, если второй — яблоко.

Сеть Хопфилда

Число выходов совпадает с **размерностью вектора-прототипа**.

На выходе — **вектор-прототип**, к которому ближе всего предъявленный входной вектор.

Персепtron Розенблатта (I)

Предыстория

Уоррен Маккаллок (Warren McCulloch) и **Уолтер Питтс** (Walter Pitts) (1943),
первая **необучаемая** искусственная нейронная сеть:

McCulloch W. S., Pitts W. A logical calculus of the ideas immanent in nervous activity
// Bull. Math. Biophys. – 1943. – Vol. 5. – pp. 115–133.

Маккаллок У. С., Питтс У. Логическое исчисление идей, относящихся к нервной
активности // В кн.: Автоматы. Сб. статей под ред. К. Э. Шеннона и
Дж. Маккарти: Пер. с англ. под ред. А. А. Ляпунова. – М.: Изд-во Иностранной
литературы, 1956. – с. 362–384.

Фрэнк Розенблattt (Frank Rosenblatt) (1958),
первая **обучаемая** искусственная нейронная сеть — **персепtron**:

Rosenblatt F. The perceptron: A probabilistic model for information storage and
organization in the brain // *Psychological Review*. – 1958. – Vol. 65. – pp. 386–408.

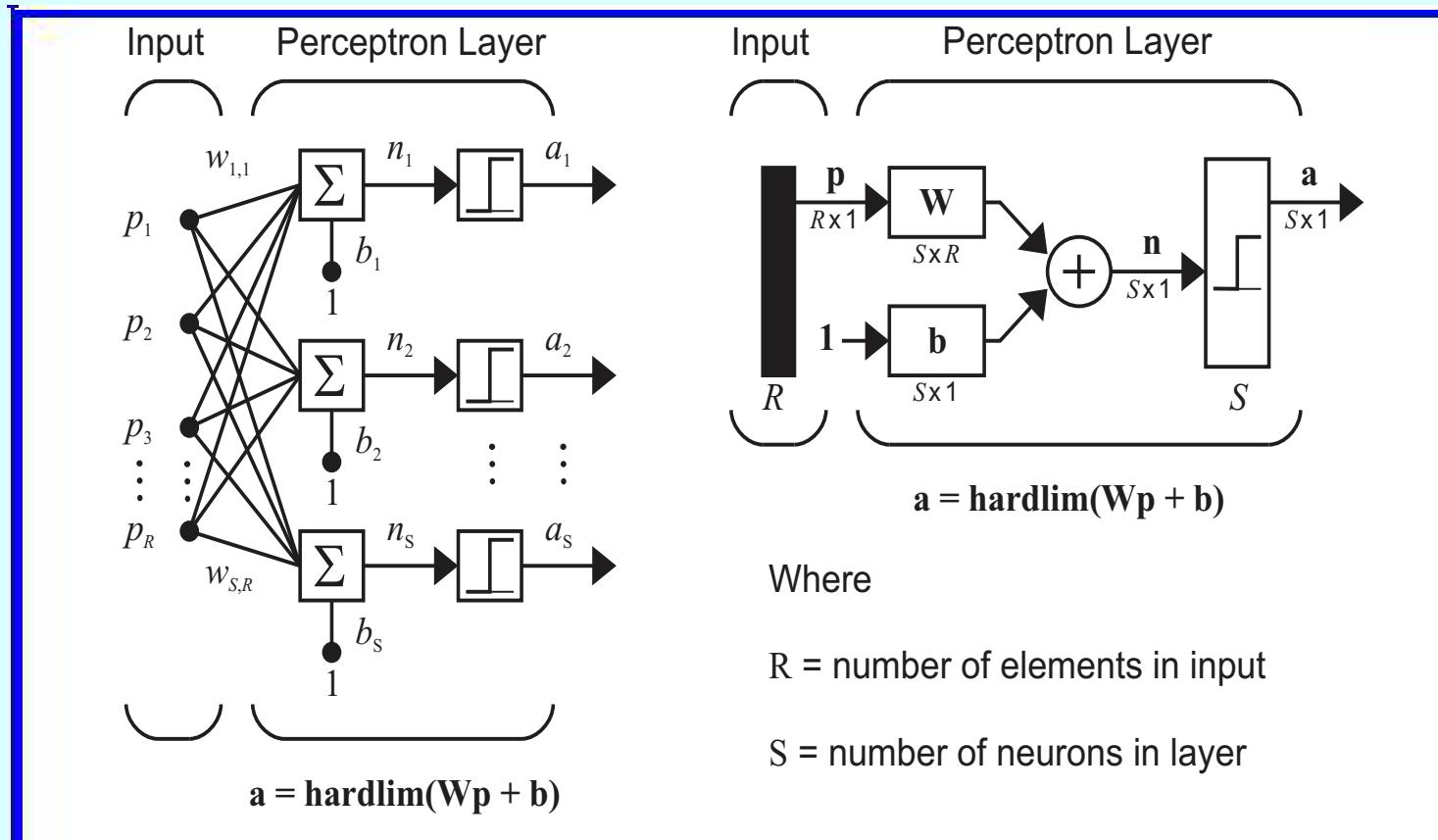
Rosenblatt F. Principles of neurodynamics: Perceptrons and the theory of brain
mechanisms. – Washington, D.C.: Spartan Books, 1962.

Розенблатт Ф. Принципы нейродинамики: Перцептроны и теория механизмов
мозга. Пер. с англ. – М.: Мир, 1965. – 480 с.

Персепtron Розенблатта (II)

Архитектура персептрана – 1

Персепtron общего вида (1)

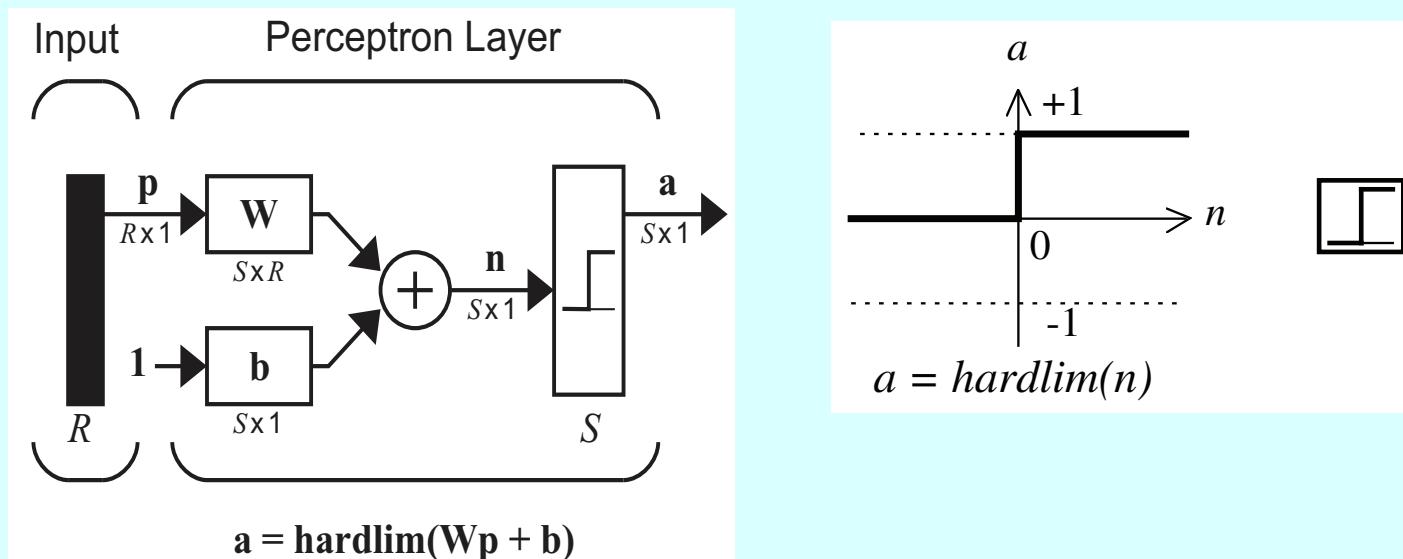


Источник: *Demuth H., Beale M., Hagan M.*. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 3-5).

Персепtron Розенблатта (III)

Архитектура персептрана – 2

Персепtron общего вида (2)

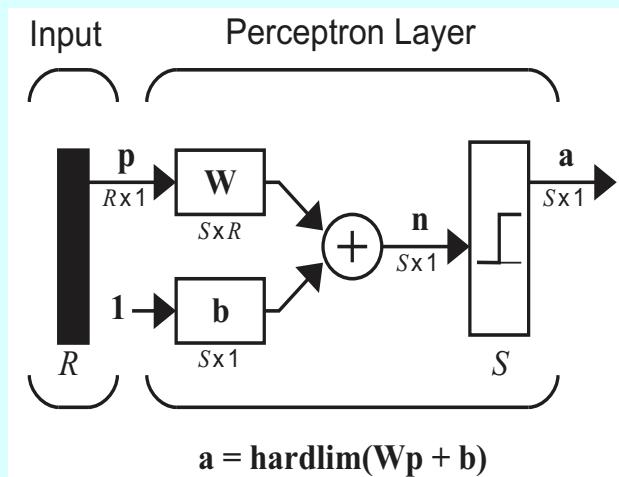


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 3-5).

Персептрон Розенблатта (IV)

Архитектура персептрана – 3

Персептрон общего вида (3)



Выход персептрана:

$$a = \text{hardlim}(W \cdot p + b)$$

Пороговая активационная функция:

$$a = \text{hardlim}(n) = \begin{cases} 1, & n \geq 0; \\ 0, & \text{в других случаях} \end{cases}$$

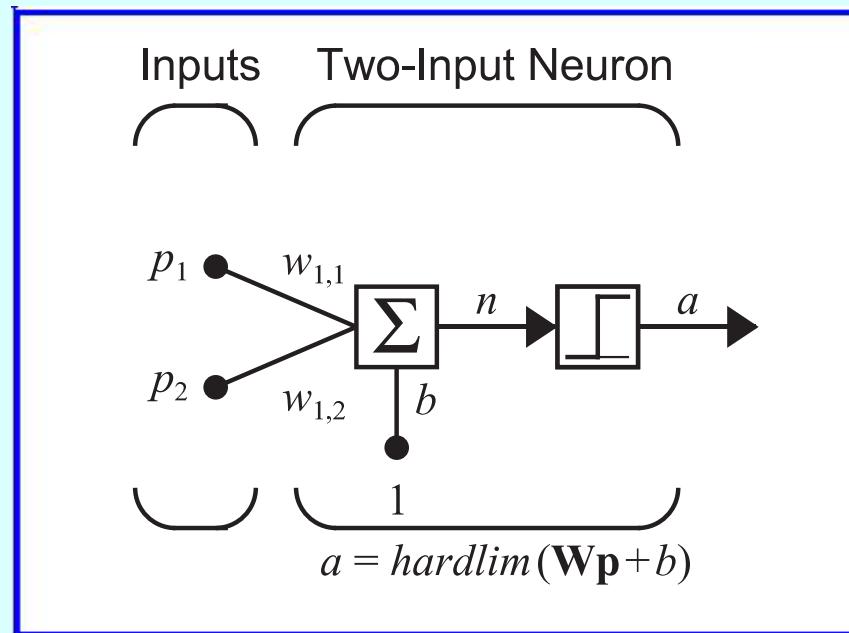
Матрица синаптических весов персептрана:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

Персептрон Розенблатта (V)

Архитектура персептрана – 4

Однонейронный персептрон (1)



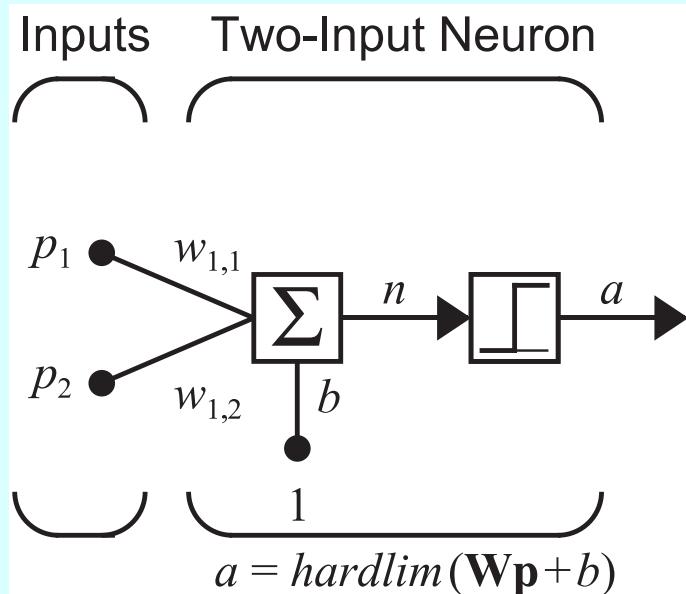
Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 4, Figure 4.2, pp.4-5).

Персепtron Розенблатта (VI)

Архитектура персептрана – 5

Однонейронный персептрон (2)

Выход персептрана:



$$\begin{aligned} a &= \text{hardlim}(W \cdot p + b) = \\ &= \text{hardlim}(w_{1,1}p_1 + w_{1,2}p_2 + b) \end{aligned}$$

Разделяющая граница:

$$n = W \cdot p + b = w_{1,1}p_1 + w_{1,2}p_2 + b = 0$$

Пример разделяющей границы:

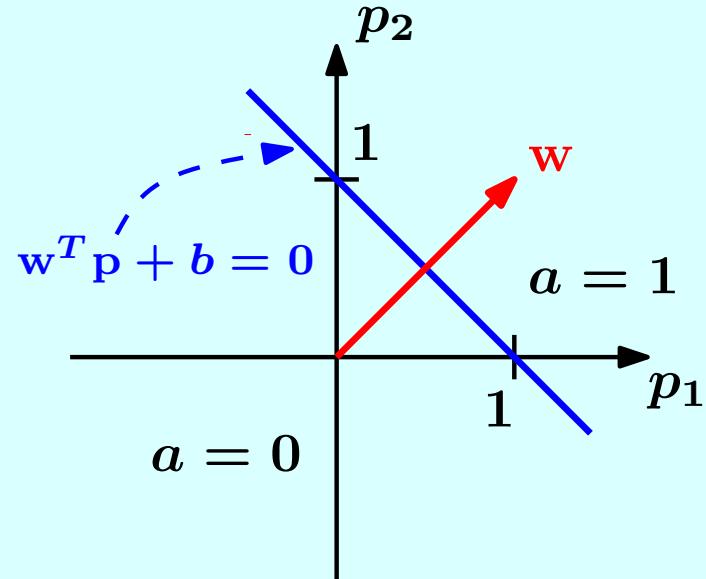
$$w_{1,1} = 1, w_{1,2} = 1, b = -1$$

$$n = w_{1,1}p_1 + w_{1,2}p_2 + b = p_1 + p_2 - 1 = 0$$

Персепtron Розенблатта (VII)

Архитектура персептрана – 6

Однонейронный персептрон (3)



Разделяющая граница:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + b = p_1 + p_2 - 1 = 0,$$

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{-1}{1} = 1, \text{ если } p_1 = 0,$$

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{-1}{1} = 1, \text{ если } p_2 = 0.$$

Вектор w ортогонален разделяющей границе персептрана.

Выход персептрана:

$$a = \text{hardlim}(n) = \text{hardlim}(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

$$a = \text{hardlim}(n) = \begin{cases} 1, & n \geq 0; \\ 0, & n < 0. \end{cases}$$

Пусть $p = [2 \ 0]^T$ – тестовый входной вектор персептрана, тогда его выход будет равен

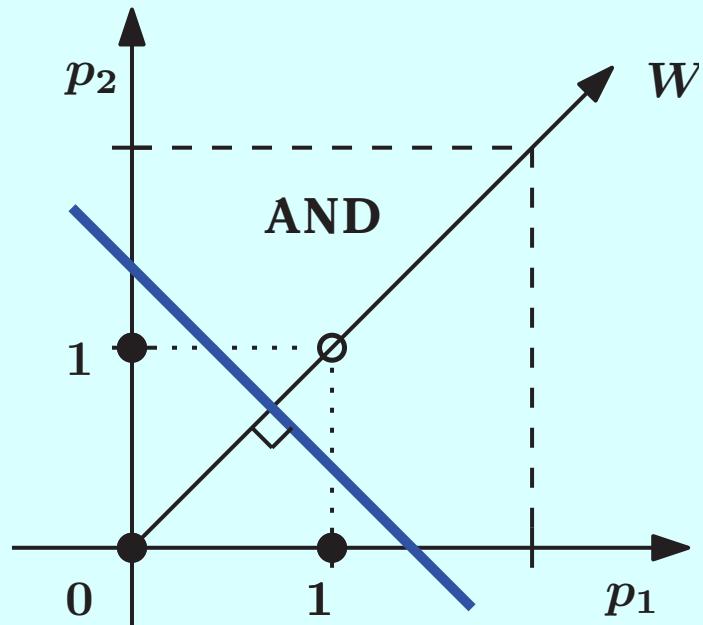
$$a = \text{hardlim}(n) = \text{hardlim}(w^T p + b) = \text{hardlim}\left([1 \ 1] \begin{bmatrix} 2 \\ 0 \end{bmatrix} - 1\right) = 1.$$

Аналогично, для $p = [0 \ 0]^T$ получим $n = -1$ и $a = \text{hardlim}(-1) = 0$.

Персепtron Розенблатта (VIII)

Архитектура персептрана – 7

Однонейронный персептрон (4)



Пример — логическая функция AND:

$$\{ p_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_1 = 0 \} \quad \{ p_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 0 \}$$

$$\{ p_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 0 \} \quad \{ p_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 1 \}$$

Построить границу между объектами двух классов («темные» и «светлые»).

Граница (синяя линия) — «примерно посередине» между объектами.

Вектор весов w — перпендикулярен границе.

Возможный выбор вектора весов:

$$w = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

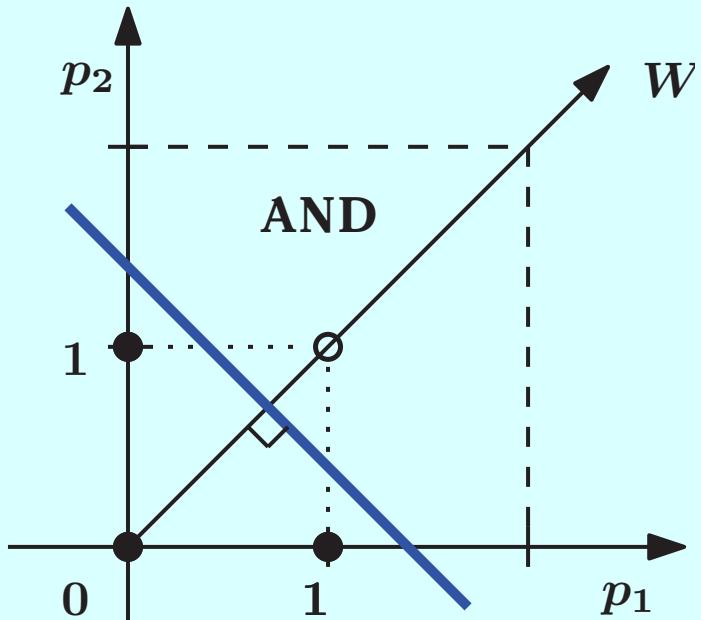
Возможный выбор смещения:

$$w^T p + b = [2 \ 2] \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} + b = 3 + b = 0 \Rightarrow b = -3.$$

Персепtron Розенблатта (IX)

Архитектура персептрана – 8

Однонейронный персептрон (5)



Пример — логическая функция AND
(проверка полученной границы между классами)

$$\begin{aligned}a_1 &= \text{hardlim}(w^T p_1 + b) = \\&= \text{hardlim}\left([2 \ 2] \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 3\right) = 0.\end{aligned}$$

$$a_2 = \text{hardlim}\left([2 \ 2] \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 3\right) = 0.$$

$$a_3 = \text{hardlim}\left([2 \ 2] \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 3\right) = 0.$$

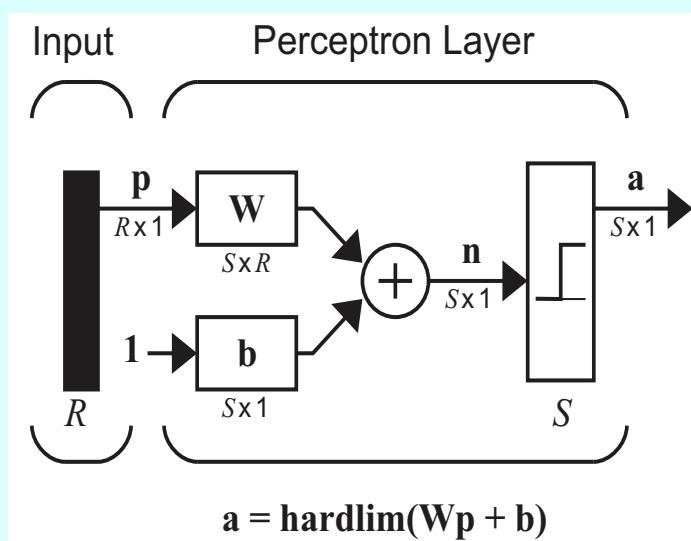
$$a_4 = \text{hardlim}\left([2 \ 2] \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 3\right) = 1.$$

Вывод: построенная граница между классами решает поставленную задачу.

Персептрон Розенблатта (X)

Архитектура персептрана – 9

Персептрон общего вида (3)



Выход персептрана:

$$a = \text{hardlim}(W \cdot p + b)$$

Разделяющая граница для i -го нейрона:

$$w_i^T \cdot p + b_i = 0,$$

$$w_i^T = [w_{i,1} \ w_{i,2} \ \dots \ w_{i,R}].$$

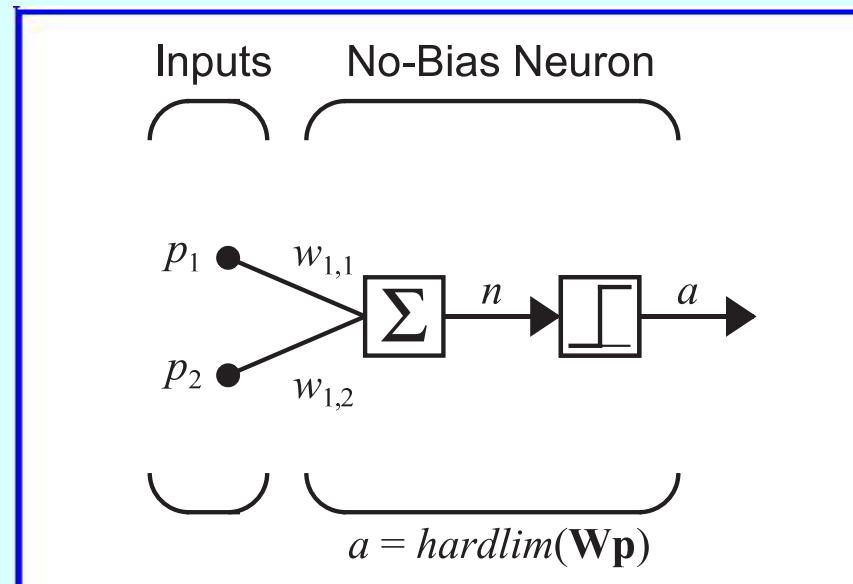
Однонейронный персептрон: один выход, принимающий значения $\{0, 1\}$, может разделять входные векторы на **два класса**.

Многонейронный персептрон: S выходов, принимающих значения $\{0, 1\}$, может разделять входные векторы на 2^S классов.

Персептрон Розенблатта (XI)

Обучающее правило персептрана – 1

Персептрон без смещения

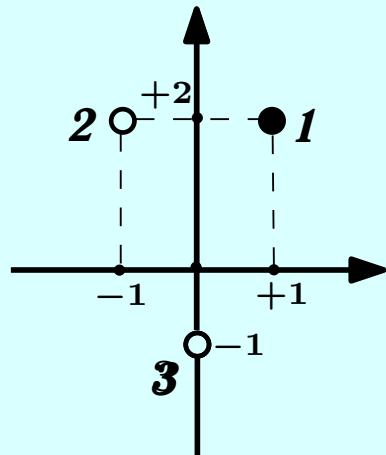


Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 4, Figure 4.4, pp.4-9).

Персепtron Розенблатта (XII)

Обучающее правило персептрана – 2

Тестовая задача

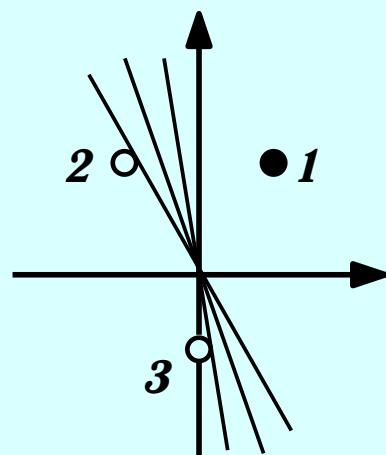


Пары вход-выход:

$$\left\{ p_1 = \begin{bmatrix} p_{1,1} \\ p_{1,2} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\}$$

$$\left\{ p_2 = \begin{bmatrix} p_{2,1} \\ p_{2,2} \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\}$$

$$\left\{ p_3 = \begin{bmatrix} p_{3,1} \\ p_{3,2} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$



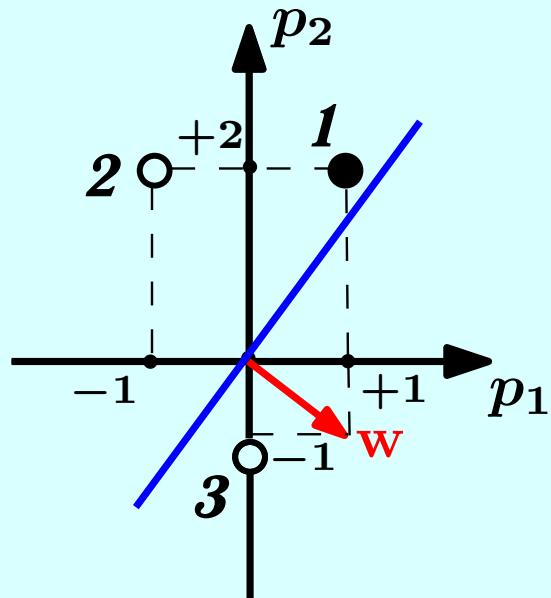
Смещение $b = 0$, поэтому разделяющая граница будет проходить всегда через начало координат.

Существует бесконечное число вариантов разделяющей границы, удовлетворяющих условию рассматриваемой задачи.

Персепtron Розенблатта (XIII)

Обучающее правило персептрана – 3

Формирование обучающего правила (1)



Начальное (произвольное) значение вектора весов сети:

$$w^T = [1.0 \ -0.8]$$

Проверка правильности классификации, выполняемой сетью с такими весами:

$$a_i = \text{hardlim}(w^T \cdot p_i), \quad i = 1, 2, 3$$

$$p_1 = [1 \ 2]^T, \quad t_1 = 1$$

$$a_1 = \text{hardlim}([1.0 \ -0.8] \begin{bmatrix} 1 \\ 2 \end{bmatrix}) = 0$$

$$p_2 = [-1 \ 2]^T, \quad t_2 = 0, \quad a_2 = 0$$

$$p_3 = [0 \ -1]^T, \quad t_3 = 0, \quad a_3 = 1$$

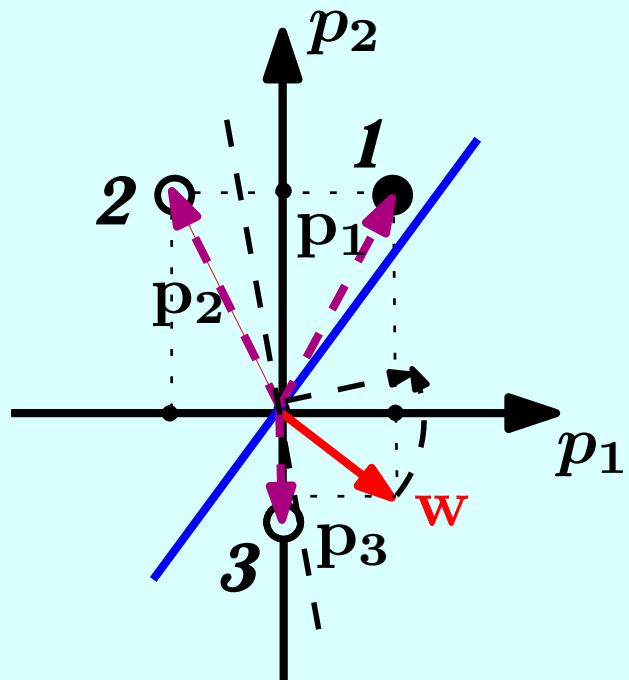
Результат: при данном принятом значении весового вектора w классификация входных векторов p_1 и p_3 выполнена **неверно**.

Требуется: изменить значение весового вектора w таким образом, чтобы **все** входные векторы p_1, p_2, p_3 классифицировались корректно.

Персепtron Розенблатта (XIV)

Обучающее правило персептрана – 4

Формирование обучающего правила (2)



Обучающее правило Розенблатта:

Общий принцип: если входной вектор p расположен «неправильно» ($a_i \neq t_i$) относительно разделяющей границы, развернуть эту границу, изменяя значение весового вектора w , так, чтобы вектор p располагался «правильно» ($a_i = t_i$).

Варианты взаимного расположения векторов w и p :

- (1) Векторы w и p образуют **тупой** или прямой угол.
- (2) Векторы w и p образуют **острый** угол.

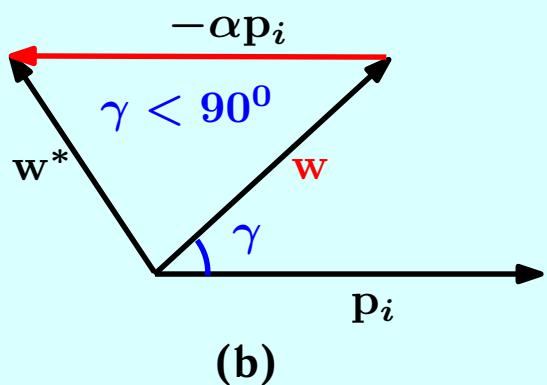
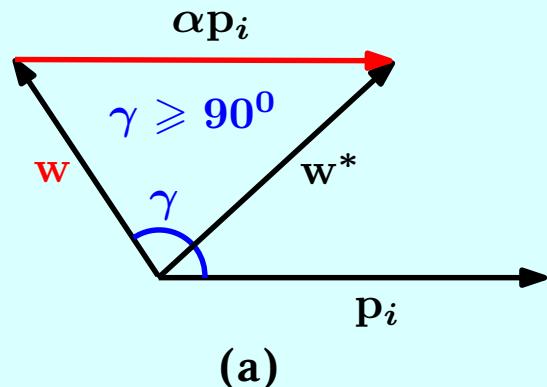
Реализация общего принципа:

- (1) Если $t = 1$ и $a = 0$, то $w^* = w + p$
- (2) Если $t = 0$ и $a = 1$, то $w^* = w - p$
- (3) Если $t = a$, то $w^* = w$

Персепtron Розенблатта (XV)

Обучающее правило персептрана – 5

Формирование обучающего правила (3)



Геометрическая интерпретация процедуры обучения персептрана:

$$a_i = \text{hardlim}(w^T \cdot p_i), \quad i = 1, 2, 3$$

$$w^T \cdot p_i = w_{1,1}p_{i,1} + w_{1,2}p_{i,2} = (w, p_i)$$

(w, p_i) – скалярное произведение векторов w и p_i

$$(w, p_i) = |w| |p_i| \cos \gamma$$

$$|w| = \sqrt{w_{1,1}^2 + w_{1,2}^2} \quad |p_i| = \sqrt{p_{i,1}^2 + p_{i,2}^2}$$

$$0 \leq \gamma < 90^\circ \Rightarrow \cos \gamma > 0$$

$$90^\circ \leq \gamma < 270^\circ \Rightarrow \cos \gamma \leq 0$$

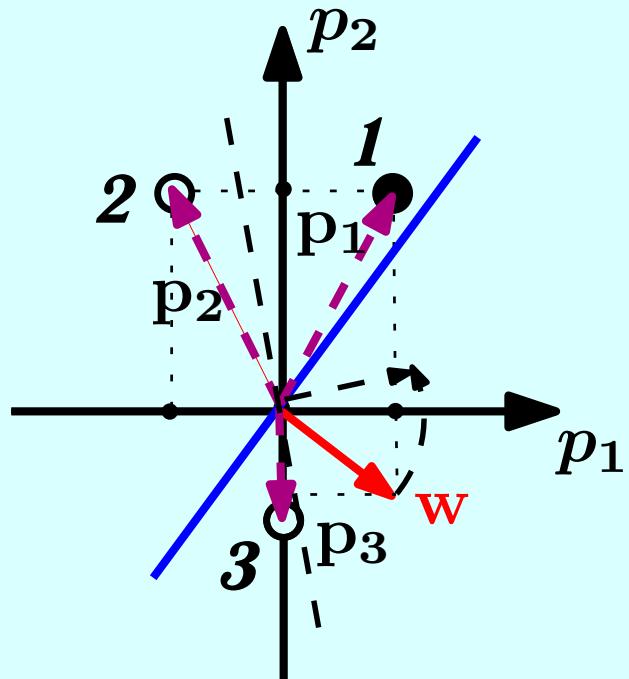
(a) $\cos \gamma \geq 0 \Rightarrow \gamma \geq 90^\circ \Rightarrow w^* = w + \alpha p_i$

(b) $\cos \gamma < 0 \Rightarrow \gamma < 90^\circ \Rightarrow w^* = w - \alpha p_i$

Персепtron Розенблатта (XVI)

Обучающее правило персептрана – 6

Формирование обучающего правила (4)



Обучающее правило Розенблатта, выраженное через ошибку классификации:

$e = t - a$ – ошибка классификации

- (1) Если $e = 1$, то $w^* = w + p$
- (2) Если $e = -1$, то $w^* = w - p$
- (3) Если $e = 0$, то $w^* = w$

Компактная форма обучающего правила Розенблатта:

$$w^* = w + ep = w + (t - a)p$$

Учет смещения b в обучающем правиле:

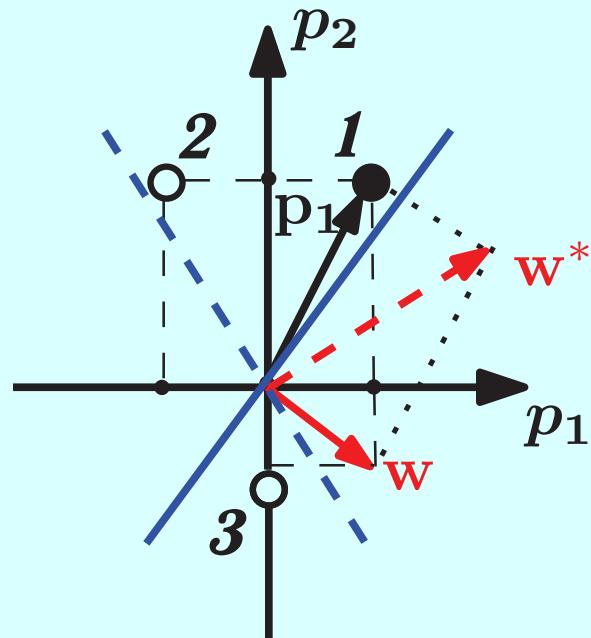
Смещение b можно считать дополнительным входным сигналом $p_0 = 1$, $w_0 = b$, тогда

$$b^* = b + e$$

Персепtron Розенблатта (XVII)

Обучающее правило персептрана – 7

Формирование обучающего правила (5)



Процесс обучения – шаг 1:

Неверно классифицирован вектор p_1 .
Угол γ между векторами w и p_1 – тупой, следовательно:

$$w^* = w + p_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$

Проверка правильности классификации:

$$p_1 = [1 \ 2]^T, \quad t_1 = 1; \quad w^T \cdot p_1 = 4.2, \quad a_1 = 1$$

$$p_2 = [-1 \ 2]^T, \quad t_2 = 0; \quad w^T \cdot p_2 = 0.4, \quad a_2 = 1$$

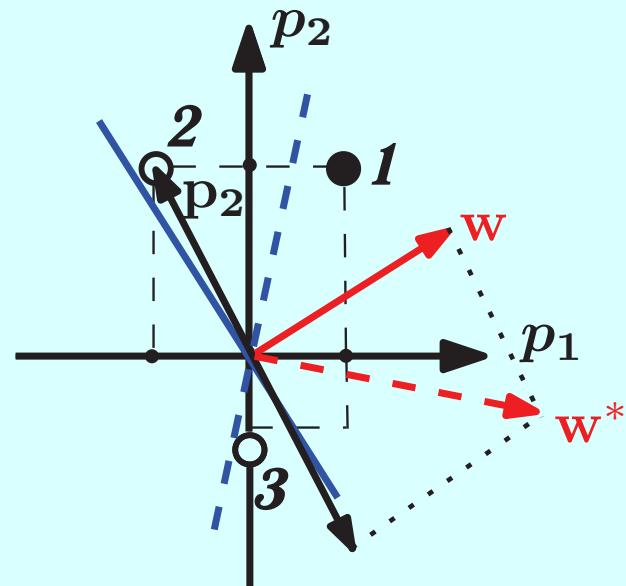
$$p_3 = [0 \ -1]^T, \quad t_3 = 0; \quad w^T \cdot p_3 = -1.2, \quad a_3 = 0$$

При $w^T = [2.0 \ 1.2]$, полученным на шаге 1, входные векторы p_1, p_3 классифицируются **правильно**, а вектор p_2 – **неправильно**.

Персептрон Розенблатта (XVIII)

Обучающее правило персептрана – 8

Формирование обучающего правила (6)



Процесс обучения – шаг 2:

Неверно классифицирован вектор p_2 .

Угол γ между векторами w и p_2 – острый, следовательно:

$$w^* = w - p_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$

Проверка правильности классификации:

$$p_1 = [1 \ 2]^T, \ t_1 = 1; \quad w^T \cdot p_1 = 0.4, \ a_1 = 1$$

$$p_2 = [-1 \ 2]^T, \ t_2 = 0; \quad w^T \cdot p_2 = -4.6, \ a_2 = 0$$

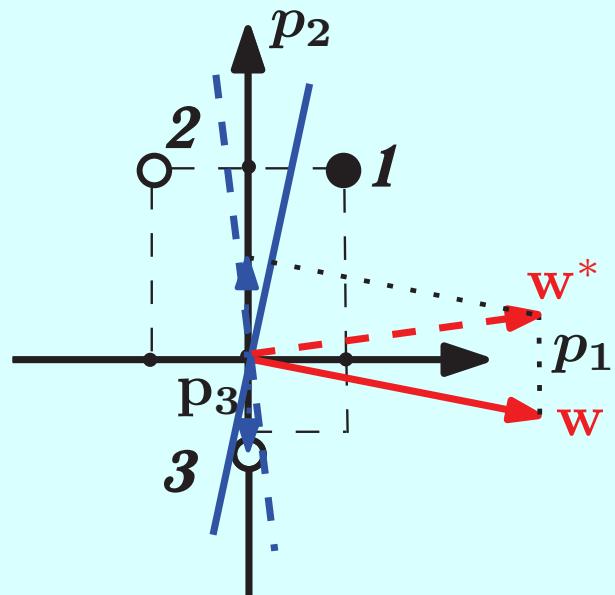
$$p_3 = [0 \ -1]^T, \ t_3 = 0; \quad w^T \cdot p_3 = 0.8, \ a_3 = 1$$

При $w^T = [3.0 \ -0.8]$, полученным на шаге 2, входные векторы p_1, p_2 классифицируются **правильно**, а вектор p_3 – **неправильно**.

Персептрон Розенблатта (XIX)

Обучающее правило персептрана – 9

Формирование обучающего правила (7)



Процесс обучения – шаг 3:

Неверно классифицирован вектор p_3 . Угол γ между векторами w и p_3 – острый, следовательно:

$$w^* = w - p_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}$$

Проверка правильности классификации:

$$p_1 = [1 \ 2]^T, \quad t_1 = 1; \quad w^T \cdot p_1 = 3.4, \quad a_1 = 1$$

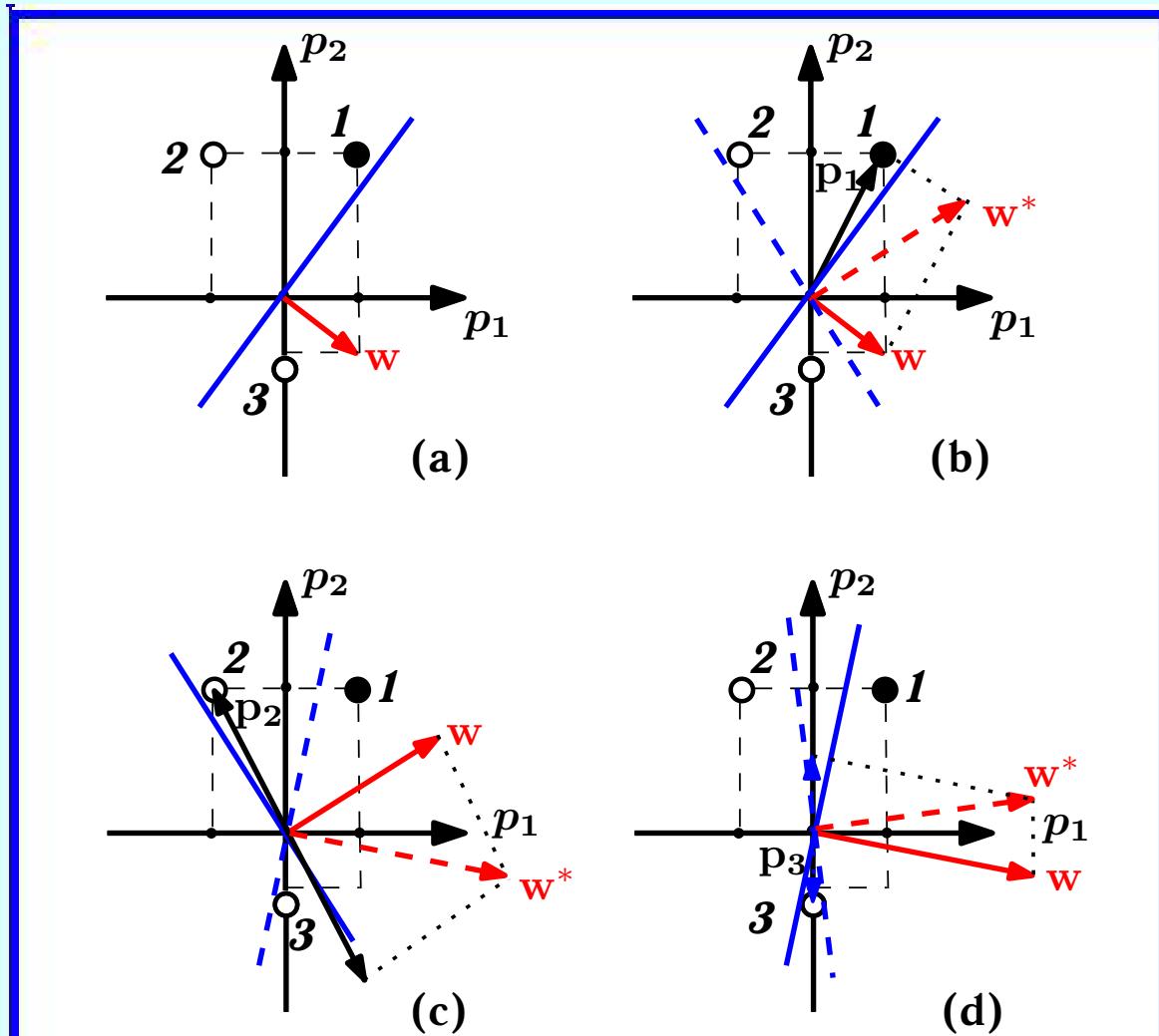
$$p_2 = [-1 \ 2]^T, \quad t_2 = 0; \quad w^T \cdot p_2 = -3.4, \quad a_2 = 0$$

$$p_3 = [0 \ -1]^T, \quad t_3 = 0; \quad w^T \cdot p_3 = -0.2, \quad a_3 = 0$$

При $w^T = [3.0 \ 0.2]$, полученным на шаге 3, входные векторы p_1, p_2, p_3 классифицируются **правильно**.

Персепtron Розенблатта (ХХ)

Обучающее правило персептрана – 10



Персепtron Розенблатта (XXI)

Обучение персептрана общего вида – 1

Объект корректировки — матрица синаптических весов \mathbf{W} :

Корректировка i -й строки матрицы \mathbf{W} :

$$\mathbf{w}_i^* = \mathbf{w}_i + e_i \mathbf{p}$$

Корректировка i -го элемента вектора смещений \mathbf{b} :

$$b_i^* = b_i + e_i$$

Правило обучения персептрана общего вида
в векторно-матричной записи:

$$\mathbf{W}^* = \mathbf{W} + \mathbf{e} \mathbf{p}^T$$

$$\mathbf{b}^* = \mathbf{b} + \mathbf{e}$$

Персепtron Розенблатта (XXII)

Обучение персептрана общего вида – 2

Процедура обучения персептрана:

1. Весовые коэффициенты \mathbf{W} и смещения \mathbf{b} сети инициализируются **случайным образом** или устанавливаются в нулевое состояние.
2. На входы сети поочередно подаются **входные векторы** \mathbf{p}_i из обучающего набора, преобразуемые в **выходные сигналы** a_i .
3. Если реакция сети a_i совпадает с эталонной t_i , т.е. $a_i = t_i$, то весовые коэффициенты \mathbf{W} и смещения \mathbf{b} **не меняются**.
4. Если реакция сети a_i не совпадает с эталонной t_i , т.е. $a_i \neq t_i$, то весовые коэффициенты \mathbf{W} **корректируются** по правилу

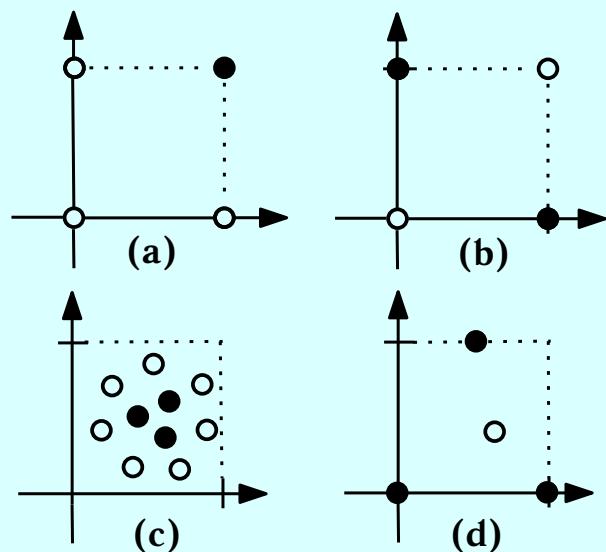
$$\mathbf{W}^* = \mathbf{W} + \mathbf{e} \mathbf{p}^T, \quad \mathbf{b}^* = \mathbf{b} + \mathbf{e}$$

5. Процедура выполняется до тех пор, пока не станет $a_i = t_i$ **для всех входных векторов** \mathbf{p}_i , или пока **не перестанут изменяться** значения весовых коэффициентов \mathbf{W} .

Ф. Розенблатт доказал **теорему сходимости** персептрана, согласно которой алгоритм его обучения сходится за **конечное** число шагов, **если существует решение** задачи.

Персептрон Розенблатта (XXIII)

Ограничения персепtronной модели – 1



Линейная разделимость классов:

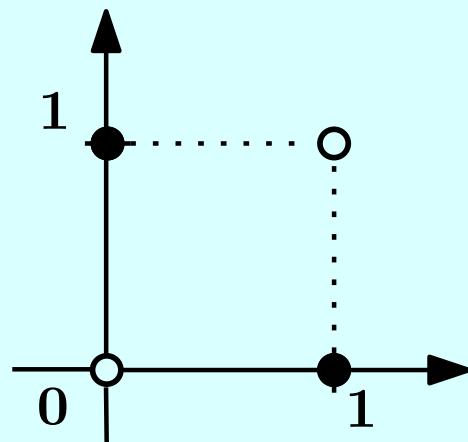
Случай (а): Классы **являются** линейно разделимыми.

Случаи (б), (в), (г): Классы **не являются** линейно разделимыми.

Персептрон **гарантирует** получение решения **только** для линейно разделимых классов.

Персептрон Розенблатта (XXIV)

Ограничения персептронной модели – 2



Логическая функция XOR (исключающее ИЛИ):

Канонический пример задачи, недоступной
для персептрана Розенблатта

$$\left\{ p_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_1 = 0 \right\} \quad \left\{ p_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 1 \right\}$$

$$\left\{ p_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 1 \right\} \quad \left\{ p_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 0 \right\}$$

Критика персептрана Розенблатта в книге:

Minsky M., Papert S. Perceptrons: An introduction to computational geometry. –
The MIT Press, 1969.

Минский М., Пейперт С. Персептроны. Пер. с англ. – М.: Мир, 1971. – 261 с.

Персептрон Розенблатта (XXV)

Ограничения персептронной модели – 3

**Свойства персептрана
и порождаемые ими фундаментальные ограничения:**

- **Однослойность** \Rightarrow Разделяющая поверхность — **гиперплоскость**.
- **Пороговая функция** в качестве активационной \Rightarrow Невозможность использования методов обучения, требующих **дифференцируемости** активационной функции (метод обратного распространения ошибки).

Обучение сетей — правило Хебба (I)

Правило обучения Хебба – 1

Дональд Хебб (Donald Hebb) (1949),
биологически правдоподобный принцип обучения нейронных сетей;
первое работающее правило обучения искусственных нейронных сетей.

Постулат Хебба

Let us assume then that the persistence or repetition of a reverberatory activity (or “trace”) tends to induce lasting cellular changes that add to its stability. The assumption * can be precisely stated as follows: *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.*

Источник: *Hebb D.O.* The organization of behavior: A neuropsychological theory. – London a.o.: Lawrence Erlbaum Associates, 2002. – 378 pp.
(с. 62 в переиздании 2002 года).

Обучение сетей — правило Хебба (II)

Правило обучения Хебба – 2

Постулат Хебба

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

Если аксон клетки А находится достаточно близко, чтобы возбуждать клетку В, и неоднократно или постоянно принимает участие в ее возбуждении, то наблюдается некоторый процесс роста или метаболических изменений в одной или обеих клетках, ведущий к увеличению эффективности А, как одной из клеток возбуждающих В.

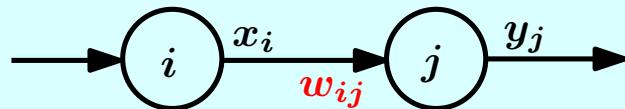
Русский перевод постулата Хебба цитируется по:

http://habrahabr.ru/blogs/artificial_intelligence/102305/

Обучение сетей – правило Хебба (III)

Правило обучения Хебба – 3

постулат Хебба \Rightarrow правило обучения Хебба



Согласно правилу Хебба **обучение происходит в результате усиления силы связи** (синаптического веса) между одновременно активными нейронами.

Часто используемые связи в сети усиливаются, что объясняет феномен обучения путем повторения и привыкания.

Правило обучения Хебба:

- основано на биологически правдоподобных предпосылках;
- является основой многих методов обучения искусственных нейронных сетей.

Обучение сетей — правило Хебба (IV)

Правило обучения Хебба – 4

Пусть имеются **два нейрона i и j** , между которыми существует **сила связи**, равная w_{ij} .

Правило Хебба в этом случае имеет вид:

$$w_{ij}(t + 1) = w_{ij}(t) + x_i y_j,$$

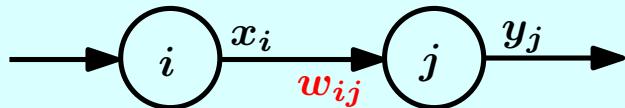
где t — время, x_i и y_j — выходы (уровни возбуждения) нейронов i и j , соответственно.

Более общая форма правила Хебба:

$$w_{ij}(t + 1) = w_{ij}(t) + \alpha x_i y_j,$$

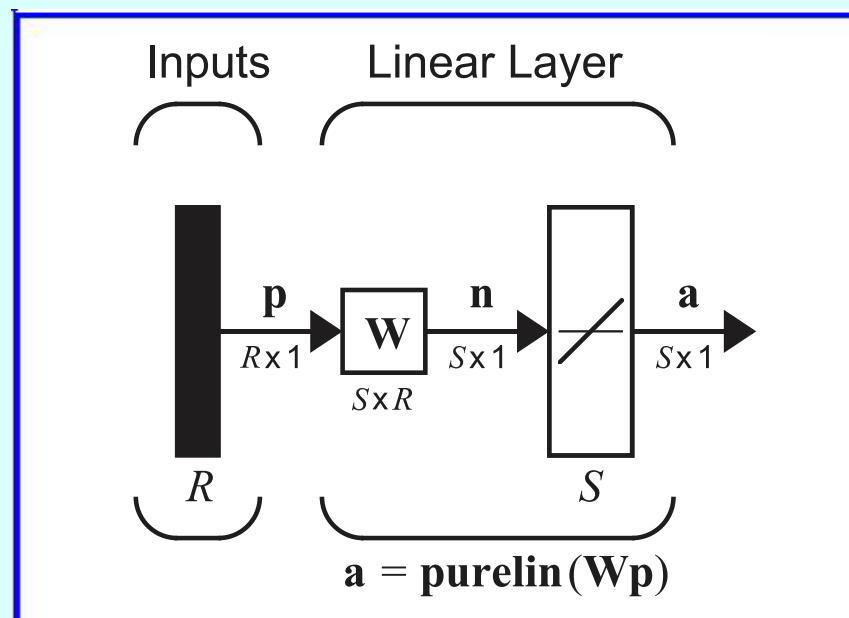
где α — коэффициент скорости обучения.

На основе правила Хебба строятся алгоритмы обучения сетей как **без учителя**, так и **с учителем**.



Обучение сетей – правило Хебба (V)

Линейный ассоциатор – 1

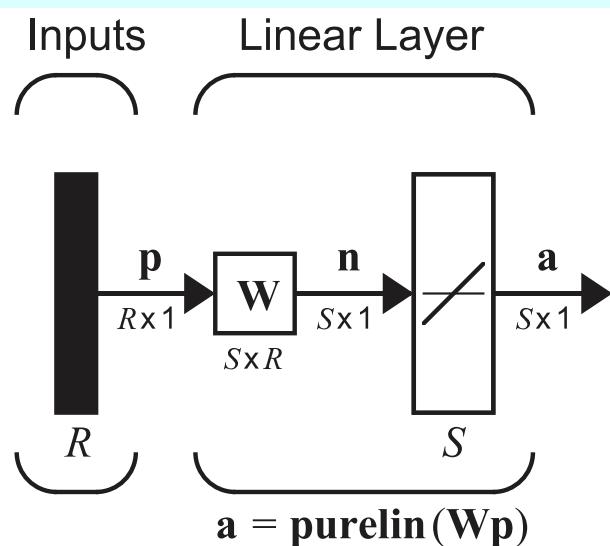


Эту сеть независимо друг от друга предложили в 1972 году **Джеймс Андерсон** (James Anderson) и **Тойво Кохонен** (Teuvo Kohonen).

Источник: *Hagan M. T., Demuth H.B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 7, Figure 7.1, pp.7-3).

Обучение сетей – правило Хебба (VI)

Линейный ассоциатор – 2



Выходной вектор \mathbf{a} в зависимости от входного вектора \mathbf{p} определяется выражением:

$$\mathbf{a} = \mathbf{W}\mathbf{p}$$

или

$$a_i = \sum_{j=1}^R w_{ij} p_j.$$

Линейный ассоциатор – пример сети, выполняющей функции **ассоциативной памяти**.

Сеть обучается на наборе пар векторов-прототипов:

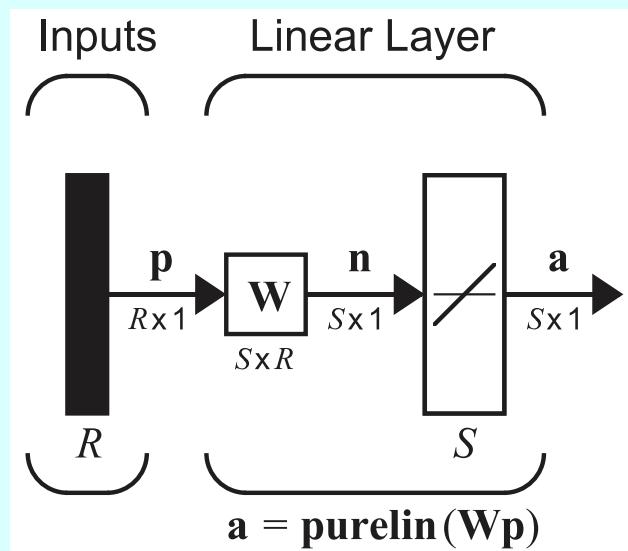
$$\{\langle \mathbf{p}_1, \mathbf{t}_1 \rangle, \langle \mathbf{p}_2, \mathbf{t}_2 \rangle, \dots, \langle \mathbf{p}_Q, \mathbf{t}_Q \rangle, \}$$

Если **обученная сеть** получает на входе вектор $\mathbf{p} = \mathbf{p}_q$, в качестве выхода она должна выдать $\mathbf{a} = \mathbf{t}_q$ для любого $q = 1, 2, \dots, Q$.

При небольшом изменении входного вектора ($\mathbf{p} = \mathbf{p}_q + \delta$), выходной вектор также должен измениться незначительно ($\mathbf{t} = \mathbf{t}_q + \varepsilon$).

Обучение сетей – правило Хебба (VII)

Линейный ассоциатор – 3



**Правило Хебба
для линейного ассоциатора:**

$$w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq},$$

где p_{jq} – j -й элемент входного вектора p_q ;
 a_{iq} – i -й элемент выходного вектора a_q , отвечающего q -му входному вектору; α – скорость обучения.

Это – правило **обучения без учителя**, оно использовано в задачах **ассоциативной памяти** и **распознавания образов**.

Обучение сетей – правило Хебба (VIII)

Линейный ассоциатор – 4

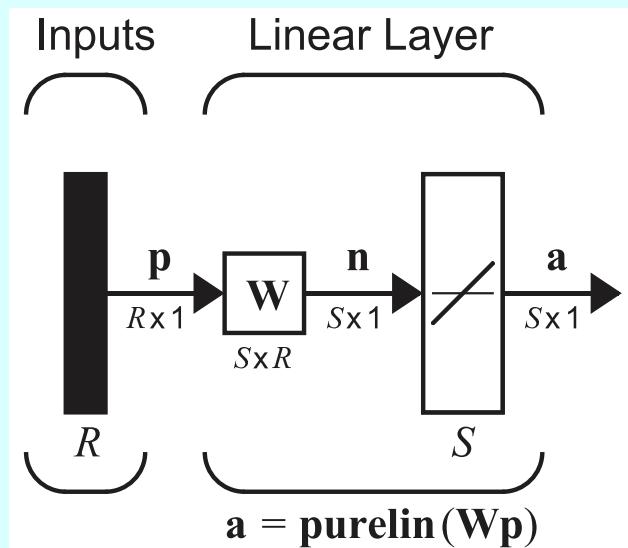
Вариант правила Хебба
для обучения с учителем:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha t_{iq} p_{jq},$$

где t_{iq} – j -й элемент выходного вектора-эталона \mathbf{t}_q .

$$\mathbf{W} = t_1 \mathbf{p}_1^T + t_2 \mathbf{p}_2^T + \dots + t_Q \mathbf{p}_Q^T = \sum_{q=1}^Q t_q \mathbf{p}_q^T,$$

$$\mathbf{W} = [\mathbf{t}_1 \ \mathbf{t}_2 \ \dots \ \mathbf{t}_Q] \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T \end{bmatrix} = \mathbf{T} \mathbf{P}^T,$$

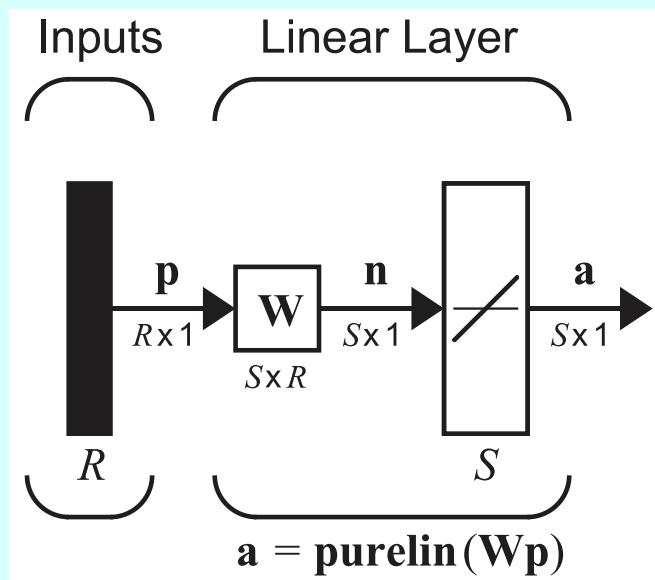


$$\mathbf{T} = [\mathbf{t}_1 \ \mathbf{t}_2 \ \dots \ \mathbf{t}_Q], \quad \mathbf{P} = [\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_Q].$$

Обучение сетей — правило Хебба (IX)

Линейный ассоциатор – 5

Свойства хеббовского обучения (1)



Выход сети a для входного вектора p_k :

$$a = Wp_k = \left(\sum_{q=1}^Q t_q p_q^T \right) p_k = \sum_{q=1}^Q t_q (p_q^T p_k)$$

Пусть входные векторы p_k ортонормированы

$$(p_q^T p_k) = 0, \quad q \neq k \quad (\text{ортогональность})$$

$$(p_q^T p_k) = 1, \quad q = k \quad (\text{нормированность})$$

В этом случае выход сети равен эталонному выходу:

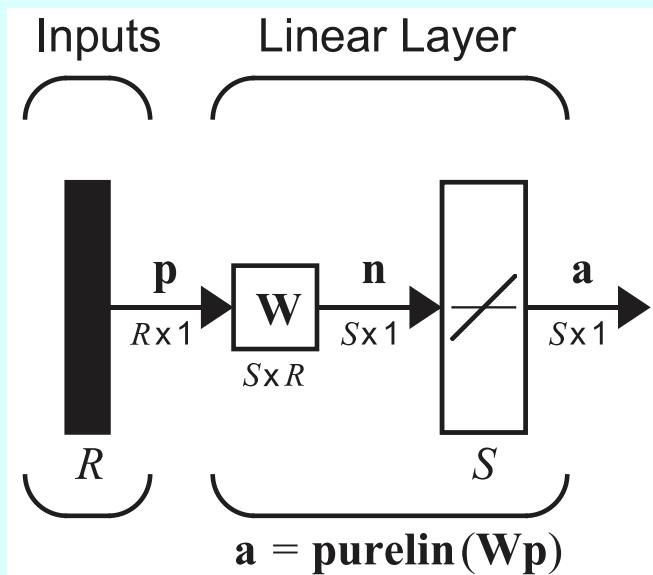
$$a = Wp_k = t_k,$$

т.е. правило Хебба в этом случае для каждого из входных векторов p_k дает правильный выход a , равный значению соответствующего вектора-эталона t_k .

Обучение сетей — правило Хебба (Х)

Линейный ассоциатор – 6

Свойства хеббовского обучения (2)



Выход сети a для входного вектора p_k в случае нормированных, но неортогональных входных векторов:

$$a = Wp_k = t_k + \sum_{q \neq k} t_q (p_q^T p_k)$$

Здесь $\sum_{q \neq k} t_q (p_q^T p_k)$ — **ошибка воспроизведения** вектора-эталона. Она увеличивается с ростом **коррелированности** входных векторов.

Обучение сетей – правило Хебба (ХI)

Линейный ассоциатор – 7

Свойства хеббовского обучения (3)

Пример 1: Ортонормированные входные векторы

$$\left\{ p_1 = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}, t_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \quad \left\{ p_2 = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}, t_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

Матрица весов для этого случая:

$$W = TP^T = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

Проверка:

$$a = Wp_1 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = t_1$$

$$a = Wp_2 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = t_2$$

Обучение сетей – правило Хебба (XII)

Линейный ассоциатор – 8

Свойства хеббовского обучения (4)

Пример 2: Нормированные, но неортогональные входные векторы
(задача о сортировке яблок и апельсинов)

Векторы-эталоны: $p_1 = [1 \ -1 \ -1]^T$ (апельсин); $p_2 = [1 \ 1 \ -1]^T$ (яблоко)

Нормированные векторы p_i : ($\tilde{p}_i = p_i / \|p_i\|$, $\|p_i\| = \sqrt{(p_i, p_i)}$, $i = 1, 2$):

$$\left\{ p_1 = \begin{bmatrix} 0.5774 \\ -0.5774 \\ -0.5774 \end{bmatrix}, \ t_1 = -1 \right\} \quad \left\{ p_2 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \ t_2 = 1 \right\}$$

Матрица весов для этого случая:

$$W = TP^T = [-1 \ 1] \begin{bmatrix} 0.5774 & -0.5774 & -0.5774 \\ 0.5774 & 0.5774 & -0.5774 \end{bmatrix} = [0 \ 1.1548 \ 0]$$

Проверка:

$$a = Wp_1 = [0 \ 1.1548 \ 0] \begin{bmatrix} 0.5774 \\ -0.5774 \\ -0.5774 \end{bmatrix} = [-0.6668] \neq t_1$$

$$a = Wp_2 = [0 \ 1.1548 \ 0] \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = [0.6668] \neq t_2$$

Обучение сетей — правило Хебба (XIII)

Линейный ассоциатор – 9

Псевдоинверсное правило обучения (1)

Линейный ассоциатор для входного вектора \mathbf{p}_q должен выдать вектор \mathbf{t}_q , т.е.

$$\mathbf{W}\mathbf{p}_q = \mathbf{t}_q, \quad q = 1, 2, \dots, Q.$$

При невозможности получить матрицу весов \mathbf{W} , обеспечивающей **точное** решение этой задачи, будем искать **приближенное** решение, например, минимизирующее **критерий** вида:

$$F(\mathbf{W}) = \sum_{q=1}^Q (\mathbf{t}_q - \mathbf{W}\mathbf{p}_q)^2$$

Если входные векторы \mathbf{p}_q **ортонормированные** и для нахождения матрицы \mathbf{W} используется **правило Хебба**, то $F(\mathbf{W}) = 0$.

В случае неортогональных векторов \mathbf{p}_q требуется найти такую матрицу \mathbf{W} , которая **минимизирует ошибку** $F(\mathbf{W})$.

Один из возможных подходов к решению этой задачи основан на использовании **псевдообратной матрицы**.

Обучение сетей – правило Хебба (XIV)

Линейный ассоциатор – 10

Псевдоинверсное правило обучения (2)

Матричная форма записи алгоритма работы линейного ассоциатора:

$$\mathbf{WP} = \mathbf{T},$$

$$\mathbf{P} = [p_1, p_2, \dots, p_Q], \quad \mathbf{T} = [t_1, t_2, \dots, t_Q]$$

Выражение для **функции ошибки** в матричном виде:

$$F(\mathbf{W}) = (\mathbf{T} - \mathbf{WP})^2 = \mathbf{E}^2, \quad \mathbf{E}^2 = \sum_i \sum_j e_{ij}^2$$

Если матрица \mathbf{P} имеет **обратную** \mathbf{P}^{-1} , веса можно найти следующим образом:

$$\mathbf{W} = \mathbf{TP}^{-1}$$

Однако в общем случае величины R (размерность векторов p_q) и Q (число векторов p_q) **не совпадают**, т.е. матрица \mathbf{P} не будет квадратной и **точного решения не существует**.

Обучение сетей — правило Хебба (XV)

Линейный ассоциатор – 11

Псевдоинверсное правило обучения (3)

Если матрица \mathbf{P} не является **квадратной**, то вместо правила $\mathbf{W} = \mathbf{T}\mathbf{P}^{-1}$ можно использовать правило

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+,$$

где \mathbf{P}^+ — псевдообратная матрица, удовлетворяющая условию $\mathbf{P}\mathbf{P}^+\mathbf{P} = \mathbf{P}$.

В случае, когда $R > Q$ (число строк в матрице \mathbf{P} больше числа столбцов), псевдообратная матрица может быть вычислена как

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

следовательно, **псевдоинверсное правило обучения** принимает вид:

$$\mathbf{W} = \mathbf{T}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$$

Обучение сетей – правило Хебба (XVI)

Линейный ассоциатор – 12

Псевдоинверсное правило обучения (4)

Пример: Сортировка яблок и апельсинов

$$\left\{ p_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = -1 \right\} \quad \left\{ p_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = 1 \right\}$$

Весовая матрица W определяется соотношением:

$$W = TP^+ = [-1 \ 1] \left(\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \right)^+$$

Псевдообратная матрица:

$$P^+ = (P^T P)^{-1} P^T = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 0.25 & -0.5 & -0.25 \\ 0.25 & 0.5 & -0.25 \end{bmatrix}$$

Весовая матрица:

$$W = TP^+ = [-1 \ 1] \begin{bmatrix} 0.25 & -0.5 & -0.25 \\ 0.25 & 0.5 & -0.25 \end{bmatrix} = [0 \ 1 \ 0]$$

Обучение сетей — правило Хебба (XVII)

Линейный ассоциатор – 13

Псевдоинверсное правило обучения (5)

Пример: Сортировка яблок и апельсинов

Весовая матрица — проверка:

$$\left\{ p_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = -1 \right\} \quad \left\{ p_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = 1 \right\}$$

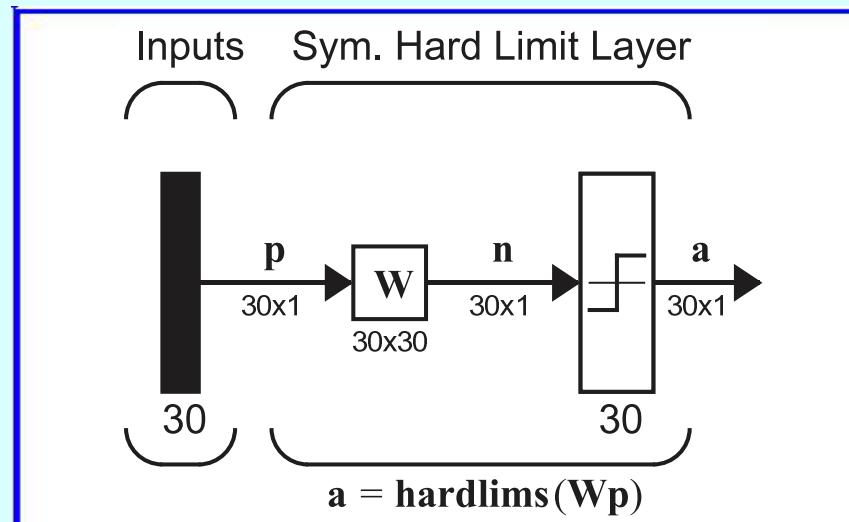
$$W = TP^+ = [-1 \ 1] \begin{bmatrix} 0.25 & -0.5 & -0.25 \\ 0.25 & 0.5 & -0.25 \end{bmatrix} = [0 \ 1 \ 0]$$

$$a = Wp_1 = [0 \ 1 \ 0] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = [-1] = t_1$$

$$a = Wp_2 = [0 \ 1 \ 0] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = [1] = t_2$$

Обучение сетей – правило Хебба (XVIII)

Автоассоциативная память – 1



В автоассоциативной памяти **требуемый выход** равен соответствующему входному вектору, т.е. $t_q = p_q$.

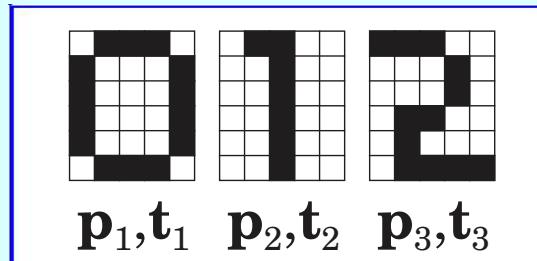
В автоассоциативную память записывается **набор паттернов** (образцов), которые затем могут быть извлечены даже в том случае, когда на вход сети подается **искаженный вектор** p_q .

Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 7, Figure 7.2, pp.7-11).

Обучение сетей — правило Хебба (XIX)

Автоассоциативная память – 2

Пример — распознавание цифр (1)



Паттерны (образцы) цифр **{0, 1, 2}** — **растровое изображение** на сетке **(6×5)**, **кодировка** — светлая клетка (-1), темная клетка ($+1$), преобразование в вектор — **по столбцам**.

Входные векторы:
(p_1, p_2, p_3 — цифры “0”, “1” и “2”, соответственно)

$$p_1 = [-1 \ 1 \ 1 \ 1 \ 1 \ -1 \ 1 \ -1 \ -1 \ -1 \ -1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ -1 \ -1 \ 1 \\ 1 \ -1 \ -1 \ -1 \ -1 \ -1 \ 1 \ -1 \ 1 \ 1 \ 1 \ 1 \ 1 \ -1] ^T$$

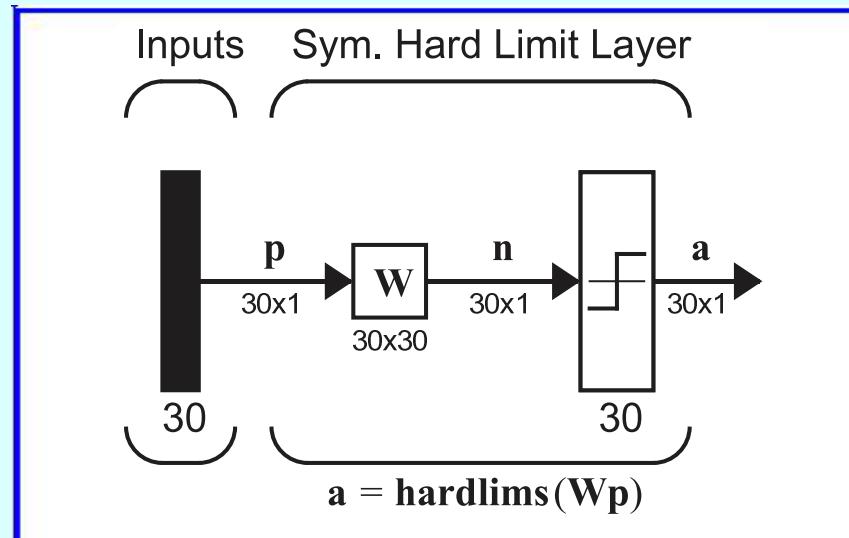
$$p_2 = [-1 \ -1 \ -1 \ -1 \ -1 \ -1 \ -11 \ -1 \ -1 \ -1 \ -1 \ -1 \ -1 \ -1 \ -1 \ -1 \ -1 \ -1 \ -1] ^T$$

$$p_3 = [1 \ -1 \ -1 \ -1 \ -1 \ -1 \ 1 \ -1 \ -1 \ 1 \ 1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ -1 \ 1 \ -1] ^T$$

Обучение сетей — правило Хебба (XVIII)

Автоассоциативная память – 3

Пример — распознавание цифр (2)



Согласно **правилу Хебба** для рассматриваемого примера (в нем $p_q = t_q$) весовая матрица **W** (размерностью (30×30)) примет вид:

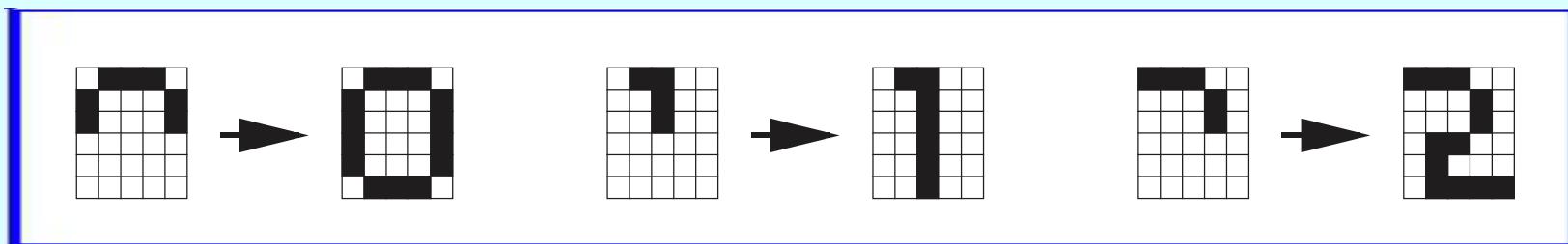
$$W = p_1 p_1^T + p_2 p_2^T + p_3 p_3^T$$

Отличие от линейного ассоциатора — пороговая функция вместо линейной на выходе сети, т.к. выходы должны принимать только два значения (растровая бинарная модель изображения).

Обучение сетей — правило Хебба (XIX)

Автоассоциативная память – 4

Пример — распознавание цифр (3)



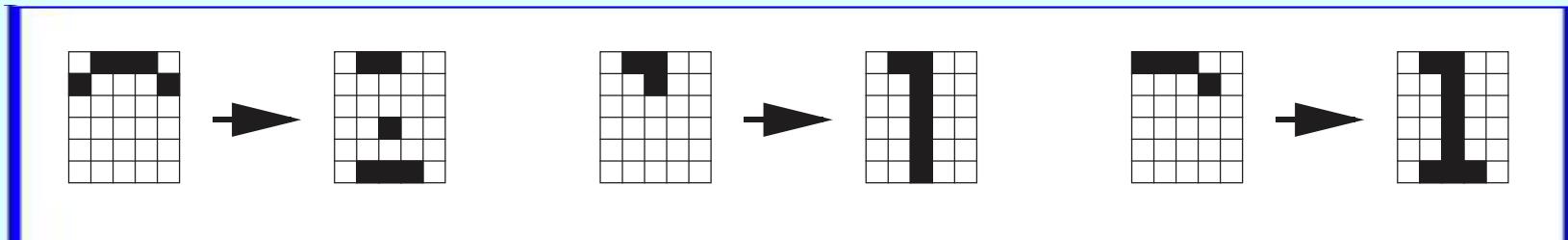
Восстановление паттернов, усеченных на 50% по высоте

Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 7, Figure 7.3, pp.7-11).

Обучение сетей — правило Хебба (ХХ)

Автоассоциативная память – 5

Пример — распознавание цифр (4)



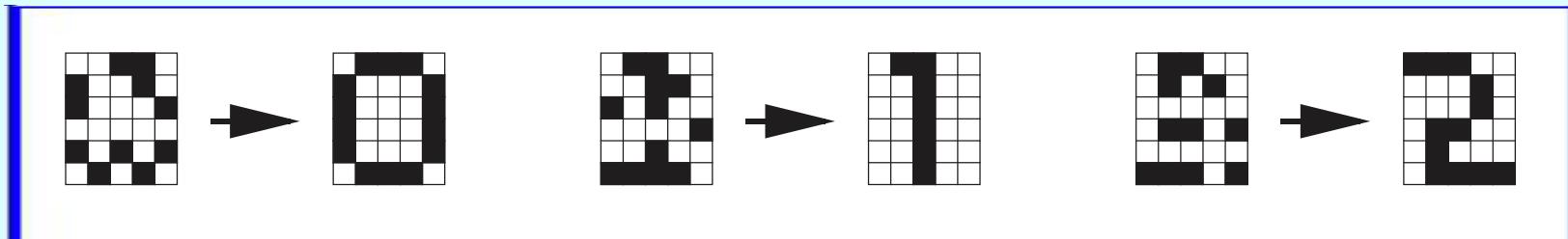
Восстановление паттернов, усеченных на 67% по высоте

Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 7, Figure 7.4, pp.7-12).

Обучение сетей — правило Хебба (XXI)

Автоассоциативная память – 6

Пример — распознавание цифр (5)

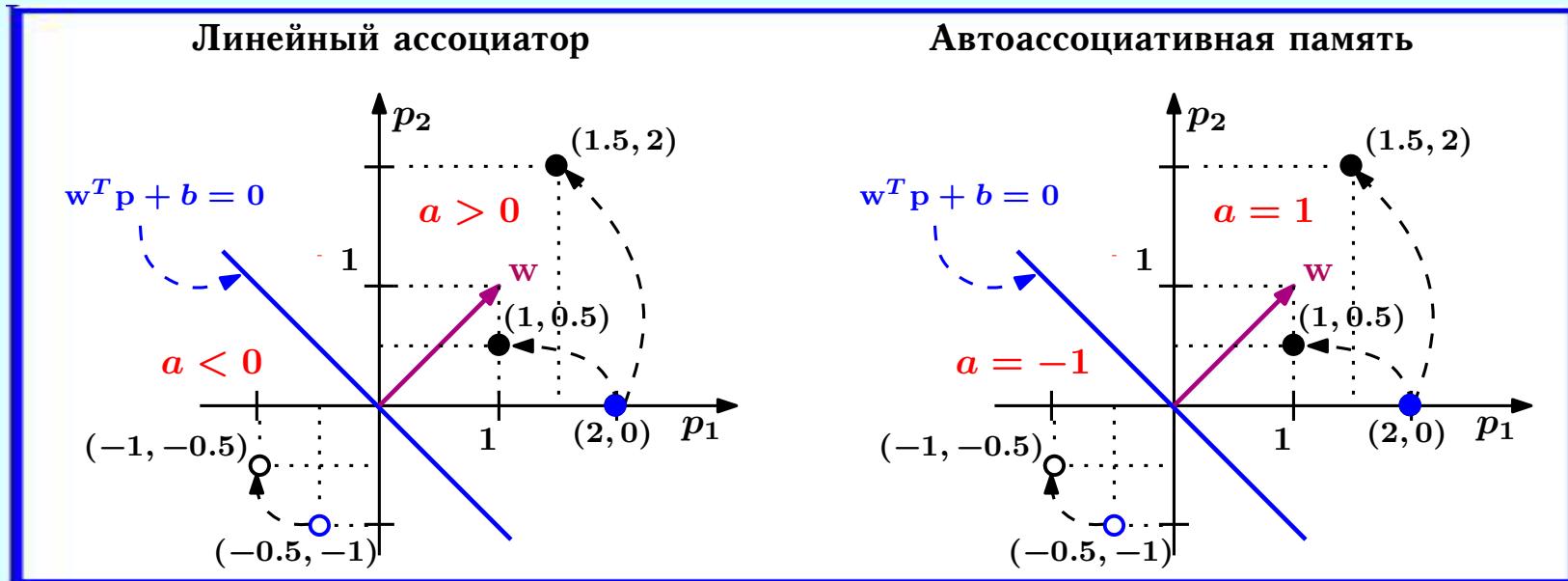


Восстановление зашумленных паттернов

Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 7, Figure 7.5, pp.7-12).

Обучение сетей – правило Хебба (ХХII)

Сравнение линейного ассоциатора и автоассоциатора – 1

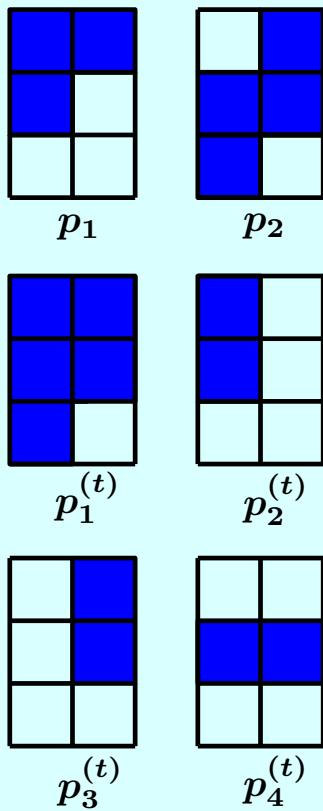


$$\begin{aligned}
 a &= \text{purelin}(n) = \\
 &= \text{purelin}(w_{1,1}p_1 + w_{1,2}p_2) = \\
 &= \text{purelin}(p_1 + p_2) \\
 a(p_1, p_2) &= w_{1,1}p_1 + w_{1,2}p_2 \\
 a(2, 0) &= 2, \quad a(-0.5, -1) = -1.5 \\
 a(1, 0.5) &= 1.5, \quad a(1.5, 2) = 3.5 \\
 a(-1, -0.5) &= -1.5
 \end{aligned}$$

$$\begin{aligned}
 a &= \text{hardlims}(n) = \\
 &= \text{hardlims}(w_{1,1}p_1 + w_{1,2}p_2) = \\
 &= \text{hardlims}(p_1 + p_2) \\
 a(p_1, p_2) &= \begin{cases} 1, & n \geq 0; \\ -1, & n < 0. \end{cases} \\
 a(2, 0) &= 1, \quad a(-0.5, -1) = -1 \\
 a(1, 0.5) &= 1, \quad a(1.5, 2) = 1 \\
 a(-1, -0.5) &= -1
 \end{aligned}$$

Обучение сетей — правило Хебба (XXIII)

Сравнение линейного ассоциатора и автоассоциатора – 2



Пример автоассоциатора (1)

Кодировка:

(+1) — темный квадрат, (-1) — светлый квадрат

Паттерны-эталоны и соответствующие им **входные векторы** p_1 и p_2 :

$$p_1 = [1 \ 1 \ -1 \ 1 \ -1 \ -1]^T; \quad p_2 = [-1 \ 1 \ 1 \ 1 \ 1 \ -1]^T$$

Векторы p_1 и p_2 **ортогональные** ($p_q^T p_k = 0$, $q \neq k$)

$$p_1^T p_2 = 0,$$

но **не нормированные** ($p_q^T p_k = 1$, $q = k$)

$$p_1^T p_1 = p_2^T p_2 = 6$$

Входные векторы для тестовых паттернов:

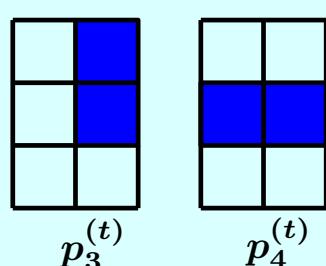
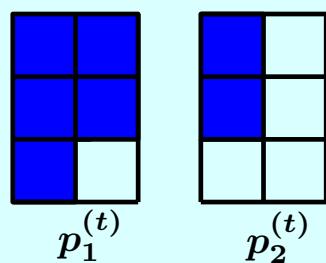
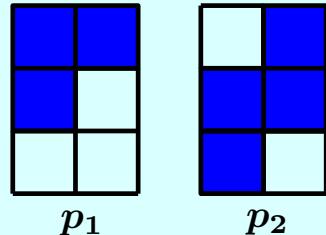
$$p_1^{(t)} = [1 \ 1 \ 1 \ 1 \ 1 \ -1]^T; \quad p_2^{(t)} = [1 \ 1 \ -1 \ -1 \ -1 \ -1]^T;$$

$$p_3^{(t)} = [-1 \ -1 \ -1 \ 1 \ 1 \ -1]^T; \quad p_4^{(t)} = [-1 \ 1 \ -1 \ -1 \ 1 \ -1]^T.$$

Обучение сетей — правило Хебба (ХХIII)

Сравнение линейного ассоциатора и автоассоциатора – 2

Пример автоассоциатора (1)



Входные векторы-эталоны:

$$p_1 = [1 \ 1 \ -1 \ 1 \ -1 \ -1]^T; \quad p_2 = [-1 \ 1 \ 1 \ 1 \ 1 \ -1]^T$$

Входные векторы для тестовых паттернов:

$$p_1^{(t)} = [1 \ 1 \ 1 \ 1 \ 1 \ -1]^T;$$

$$p_2^{(t)} = [1 \ 1 \ -1 \ -1 \ -1 \ -1]^T;$$

$$p_3^{(t)} = [-1 \ -1 \ -1 \ 1 \ 1 \ -1]^T;$$

$$p_4^{(t)} = [-1 \ 1 \ -1 \ -1 \ 1 \ -1]^T.$$

Расстояние по Хеммингу $\rho_H(p_i, p_j)$ между символьными строками p_i и p_j — количество **несовпадающих** символов в них:

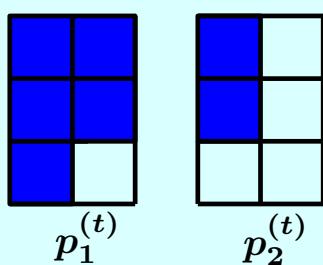
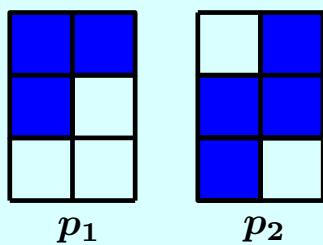
$$\rho_H(p_1, p_2) = 3, \quad \rho_H(p_1, p_1^{(t)}) = 2, \quad \rho_H(p_1, p_2^{(t)}) = 2,$$

$$\rho_H(p_1, p_3^{(t)}) = 3, \quad \rho_H(p_1, p_4^{(t)}) = 3, \quad \rho_H(p_2, p_1^{(t)}) = 1,$$

$$\rho_H(p_2, p_2^{(t)}) = 4, \quad \rho_H(p_2, p_3^{(t)}) = 2, \quad \rho_H(p_2, p_4^{(t)}) = 2$$

Обучение сетей — правило Хебба (XXIV)

Сравнение линейного ассоциатора и автоассоциатора – 3



Пример автоассоциатора (2)

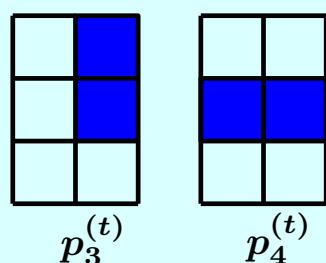
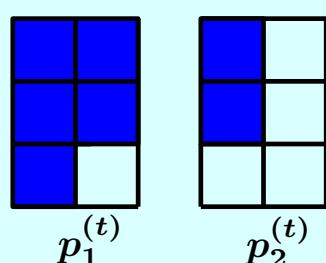
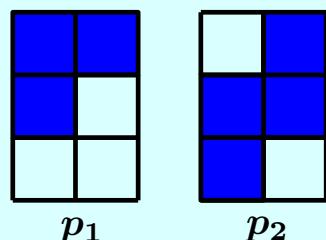
Весовая матрица согласно правилу Хебба:

$$W = TP^T, \quad T = P$$

$$\begin{aligned} W = TP^T &= \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 \end{bmatrix} = \\ &= \begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix} \end{aligned}$$

Обучение сетей — правило Хебба (XXV)

Сравнение линейного ассоциатора и автоассоциатора – 4



Пример автоассоциатора (3)

Отклик сети на 1-й тестовый паттерн ($p_1^{(t)}$):

$$a = \text{hardlims}(W p_1^{(t)}) =$$

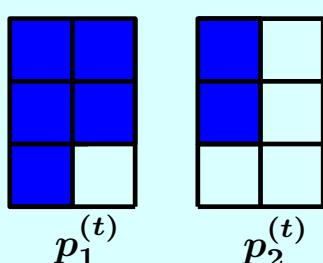
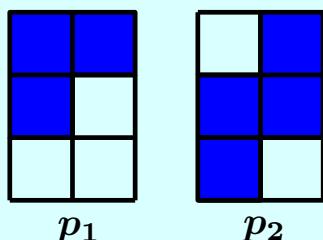
$$= \text{hardlims} \left(\begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \right) =$$

$$a = \text{hardlims} \left(\begin{bmatrix} -2 \\ 6 \\ 2 \\ 6 \\ 2 \\ -6 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = p_2$$

$$\rho_H(p_1, p_1^{(t)}) = 2, \quad \rho_H(p_2, p_1^{(t)}) = 1$$

Обучение сетей — правило Хебба (XXVI)

Сравнение линейного ассоциатора и автоассоциатора – 5



Пример автоассоциатора (4)

Отклик сети на 2-й тестовый паттерн ($p_2^{(t)}$):

$$a = \text{hardlims}(W p_2^{(t)}) =$$

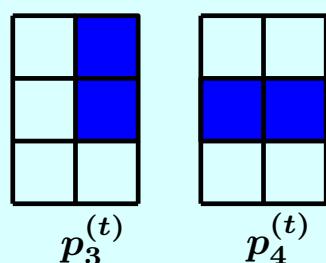
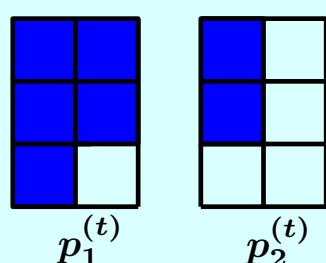
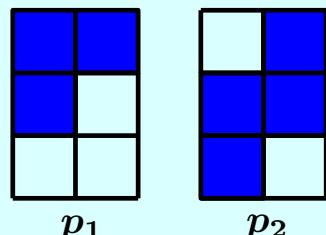
$$= \text{hardlims} \left(\begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \right) =$$

$$a = \text{hardlims} \left(\begin{bmatrix} 6 \\ 2 \\ -6 \\ 2 \\ -6 \\ -2 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ -1 \end{bmatrix} = p_1$$

$$\rho_H(p_1, p_2^{(t)}) = 2, \quad \rho_H(p_2, p_2^{(t)}) = 4$$

Обучение сетей — правило Хебба (XXVII)

Сравнение линейного ассоциатора и автоассоциатора – 6



Пример автоассоциатора (5)

Отклик сети на 3-й тестовый паттерн ($p_3^{(t)}$):

$$a = \text{hardlims}(W p_3^{(t)}) =$$

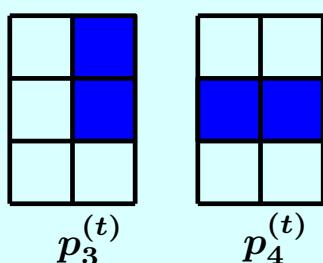
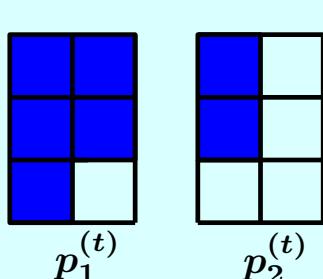
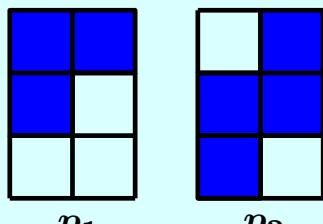
$$= \text{hardlims} \left(\begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \right) =$$

$$a = \text{hardlims} \left(\begin{bmatrix} -2 \\ 2 \\ 2 \\ 2 \\ 2 \\ -2 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = p_2$$

$$\rho_H(p_1, p_3^{(t)}) = 3, \quad \rho_H(p_2, p_3^{(t)}) = 2$$

Обучение сетей — правило Хебба (XXVIII)

Сравнение линейного ассоциатора и автоассоциатора – 7



Пример автоассоциатора (6)

Отклик сети на 4-й тестовый паттерн ($p_4^{(t)}$):

$$a = \text{hardlims}(W p_4^{(t)}) =$$

$$= \text{hardlims} \left(\begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \right) =$$

$$a = \text{hardlims} \left(\begin{bmatrix} -2 \\ 2 \\ 2 \\ 2 \\ 2 \\ -2 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = p_2$$

$$\rho_H(p_1, p_4^{(t)}) = 3, \quad \rho_H(p_2, p_4^{(t)}) = 2$$

Обучение сетей – правило Хебба (XXIX)

Варианты правила Хебба – 1

Основной вариант правила Хебба:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + t_q \mathbf{p}_q^T$$

Возможные проблемы: Элементы весовой матрицы \mathbf{W} могут принимать **очень большие значения**, если обучающий набор содержит много паттернов (векторов-образцов).

Для **ограничения** роста значений элементов матрицы \mathbf{W} в основной вариант правила Хебба можно ввести числовой параметр $\alpha > 0$ (**скорость обучения**).

Обучение сетей — правило Хебба (ХХХ)

Варианты правила Хебба – 2

Вариант правила Хебба с изменяемой скоростью обучения:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha t_q \mathbf{p}_q^T$$

В основной вариант правила Хебба вводится **скорость обучения** — числовой параметр $0 < \alpha < 1$, позволяющий **ограничить** рост значений элементов матрицы \mathbf{W} .

Дополнительно в правило Хебба можно ввести член, имитирующий **забывание** (затухание) ранее введенных входных сигналов.

Обучение сетей — правило Хебба (XXXI)

Варианты правила Хебба – 3

Вариант правила Хебба с забыванием:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha t_q \mathbf{p}_q^T - \gamma \mathbf{W}^{old} = (1 - \gamma) \mathbf{W}^{old} + \alpha t_q \mathbf{p}_q^T$$

Член, имитирующий **забывание** (затухание) ранее введенных входных сигналов вводится в правило Хебба, играя роль **сглаживающего фильтра** для входных данных.

Здесь $0 < \gamma < 1$ — **скорость забывания**.

При $\gamma \rightarrow 0$ правило Хебба сводится к его **стандартному варианту** с переменной скоростью обучения.

При $\gamma \rightarrow 1$ правило Хебба **быстро забывает** ранее введенные сигналы и помнит только паттерны, поступившие на вход сети **совсем недавно**.

Использование **забывания** в правиле Хебба позволяет **предотвратить неограниченный рост** значений элементов весовой матрицы \mathbf{W} .

Обучение сетей — правило Хебба (XXXII)

Варианты правила Хебба – 4

Вариант правила Хебба, учитывающий фактический выход сети:

$$W^{new} = W^{old} + \alpha(t_q - a)p_q^T$$

Здесь в стандартной формулировке правила Хебба **желаемый выход** сети t_q для паттерна p_q заменен на $(t_q - a)$, где a — **фактический выход** сети для того же самого паттерна при данном текущем значении весов W .

Этот вариант правила именуют часто «**дельта-правило**» или «**правило Уидроу-Хоффа**».

Дельта-правило дает результаты обучения, аналогичные **псевдоинверсному правилу**.

Псевдоинверсное правило: может быть применено, только когда известны все пары $\langle p_q, t_q \rangle$, составляющие **обучающее множество**.

Дельта-правило: позволяет модифицировать веса сети по-отдельности для каждого примера $\langle p_q, t_q \rangle$, т.е. дельта-правило пригодно **для последовательного обучения** сети в изменяющейся среде.

Обучение сетей — правило Хебба (XXXIII)

Варианты правила Хебба – 5

Вариант правила Хебба для обучения без учителя:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{a} \mathbf{p}_q^T$$

Здесь в стандартной формулировке правила Хебба **желаемый выход** сети \mathbf{t}_q для паттерна \mathbf{p}_q заменен на **a** — **фактический выход** сети для того же самого паттерна при данном текущем значении весов **W**.

Вариант правила Хебба для обучения без учителя, не требующий знать желаемые значения выходов сети, является **более точной трактовкой постулата Хебба**, чем вариант обучения с учителем.

Обучение сетей — правило Уидроу-Хоффа (I)

Предыстория

Бернард Уидроу (Bernard Widrow) — один из первых исследователей в области искусственных нейронных сетей (со второй половины 1950-х гг., как и Фрэнк Розенблatt).

Тед Хофф (Marcian “Ted” Hoff) — в 1960 году аспирант Б. Уидроу.

В 1960 году Б.Уидроу и М.Хофф предложили НС-модель **ADALINE** (ADAptive LInear NEuron или, по другому толкованию, ADAptive LINear Element), а также обучающее правило, которое они назвали алгоритмом **LMS** (Least Mean Square).

Б. Уидроу в начале 1960-х годов перестал заниматься НС-моделями и переключился на исследования в области **адаптивной обработки сигналов**, в том числе и потому, что не смог решить проблему обучения многослойных сетей. Он вернулся в нейросетевую область в начале 1980-х годов и стал заниматься применением нейросетей в **адаптивном управлении**, а также развивать метод обратного распространения во времени (*temporal backpropagation*) как развитие алгоритма LMS.

Т. Хофф с начала 1960-х годов целиком переключился на работу в области **полупроводниковой микроэлектроники**, автор идеи микропроцессора, один из основных разработчиков первого коммерчески доступного **микропроцессора Intel 4004**.

Обучение сетей — правило Уидроу-Хоффа (II)

Сеть ADALINE – 1

Сеть **ADALINE** в целом подобна **персепtronу Розенблатта**, за исключением активационных функций в них (**пороговая** функция в персептроне, **линейная** — в ADALINE).

Обучающее правило LMS более эффективное, чем правило обучения персептрана.

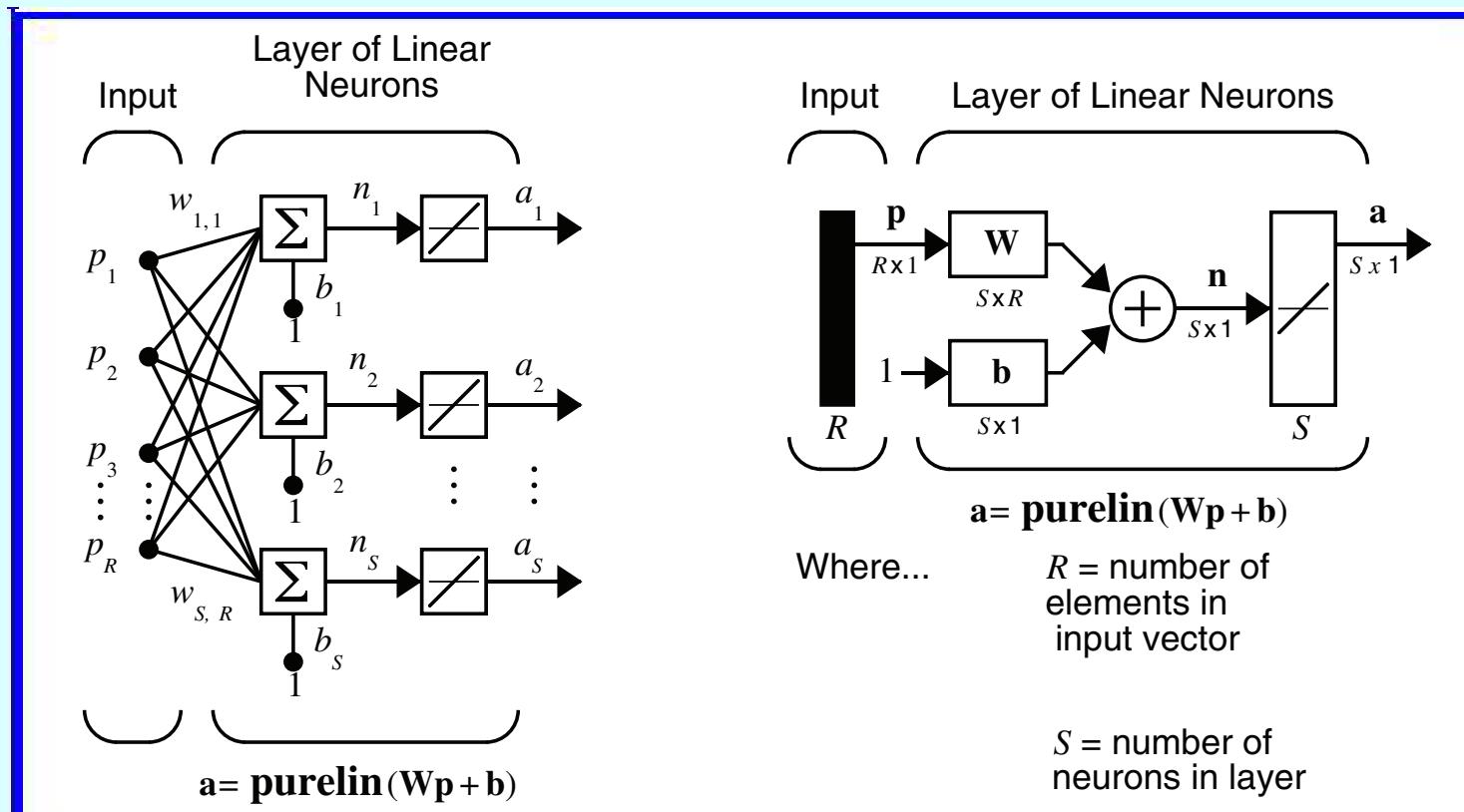
Общее ограничение персептрана и ADALINE — они могут работать только с **линейно разделимыми классами**.

ADALINE и LMS — **много применений** в области цифровой обработки данных, например, для эхоподавления в линиях дальней телефонной связи.

Обучение сетей — правило Уидроу-Хоффа (III)

Сеть ADALINE – 2

Общая структура ADALINE (1)

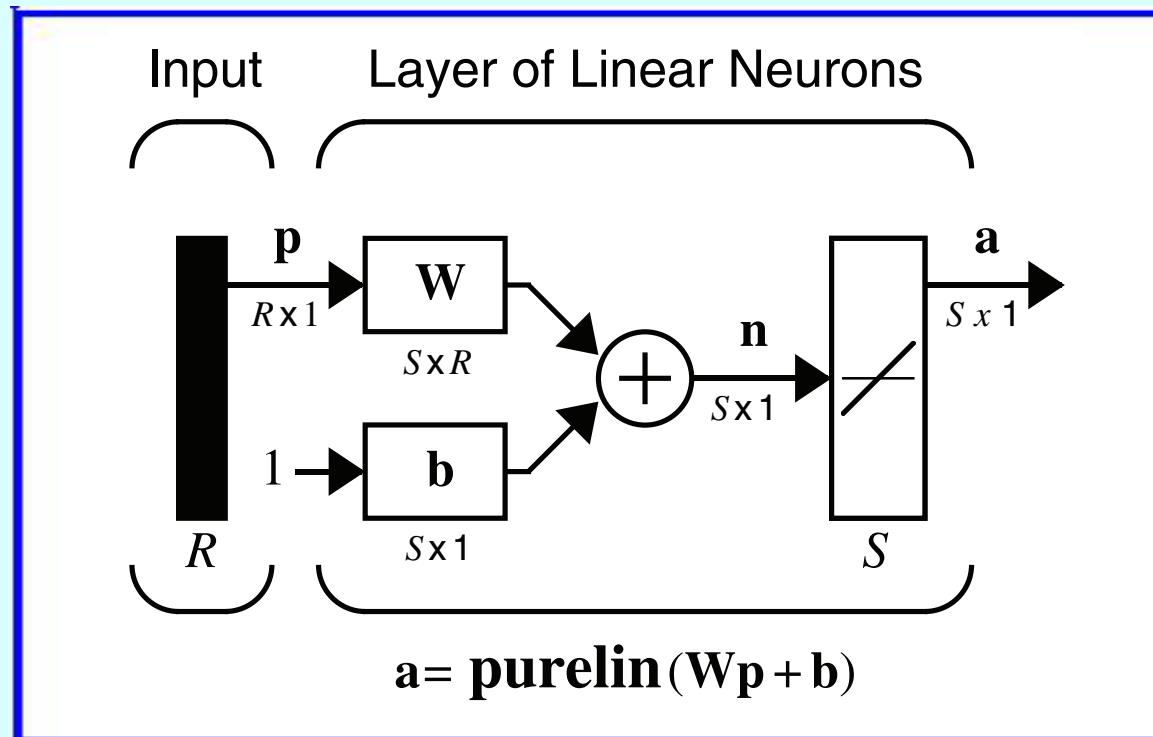


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 4-4).

Обучение сетей — правило Уидроу-Хоффа (IV)

Сеть ADALINE – 3

Общая структура ADALINE

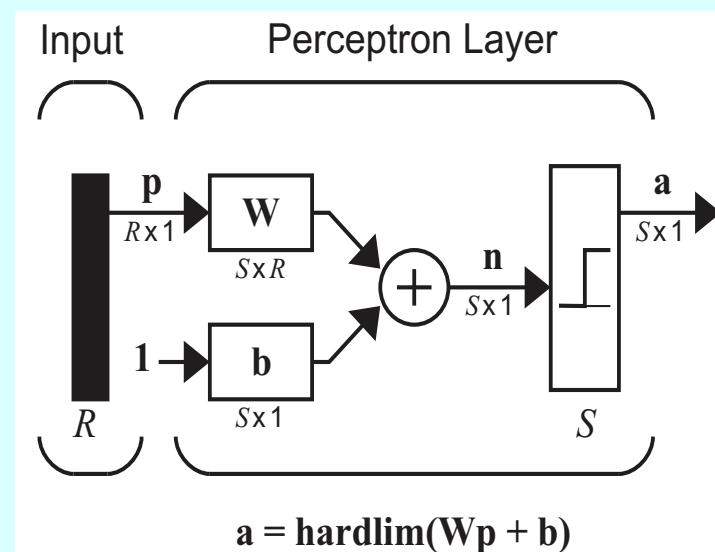
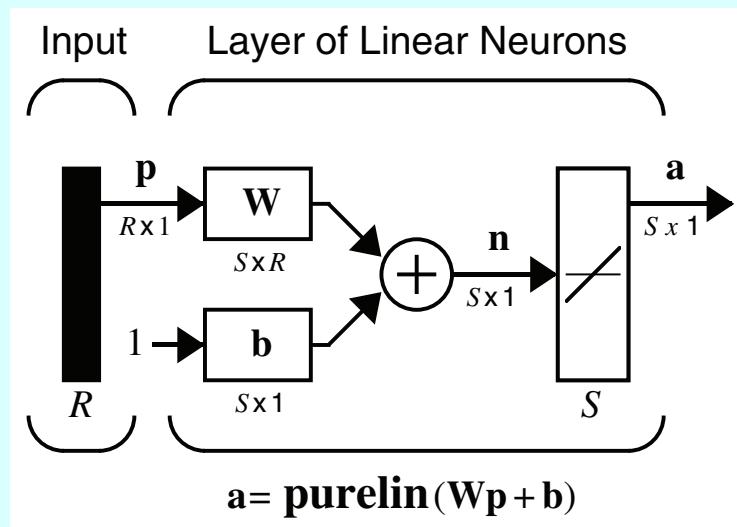


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 4-4).

Обучение сетей — правило Уидроу-Хоффа (V)

Сеть ADALINE – 4

Сравнение ADALINE и персептрана



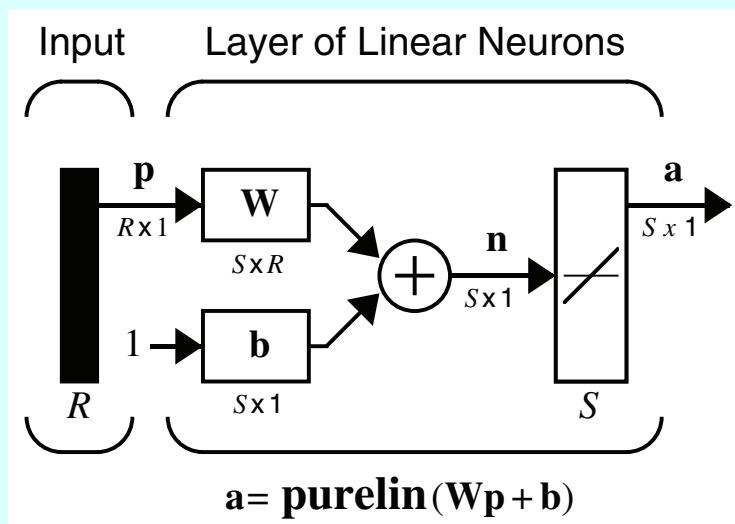
ADALINE

Персептрон

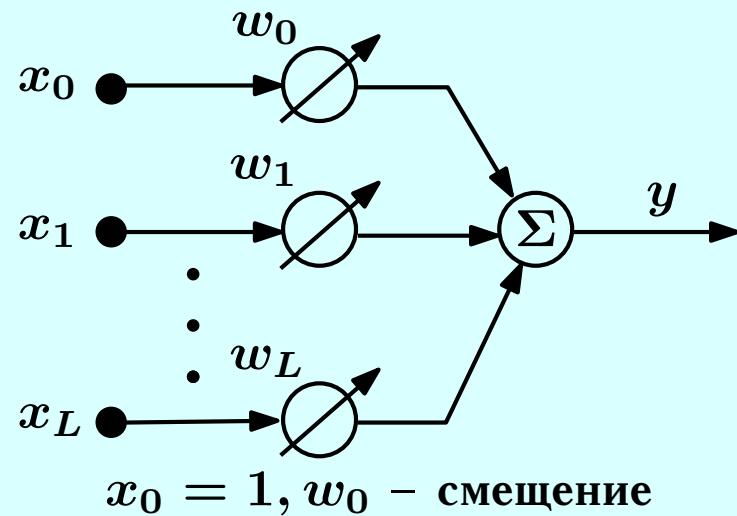
Обучение сетей — правило Уидроу-Хоффа (VI)

Сеть ADALINE – 5

Сравнение ADALINE и адаптивного линейного сумматора



ADALINE



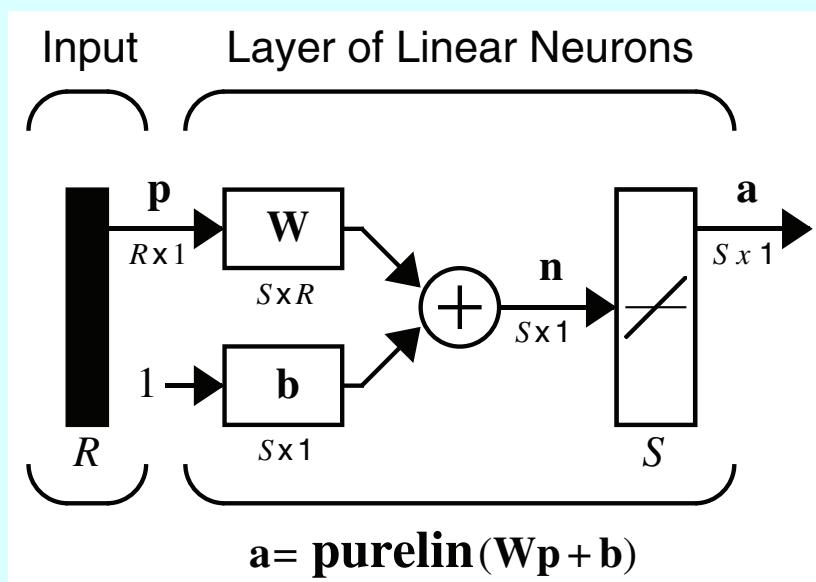
Adaptive Linear Combiner

Источник (адаптивный линейный сумматор): [Widrow B., Stearns S.D.](#) Adaptive signal processing. – Prentice-Hall, Inc. – 1985. – 474 pp. (Figure 2.1, p. 16).

Обучение сетей — правило Уидроу-Хоффа (VII)

Сеть ADALINE – 6

Алгоритм работы ADALINE



Выход сети:

$$a = \text{purelin}(Wp + b) = Wp + b$$

$$a_i = \text{purelin}(n_i) =$$

$$= \text{purelin}(w_i^T p + b_i) = w_i^T p + b_i$$

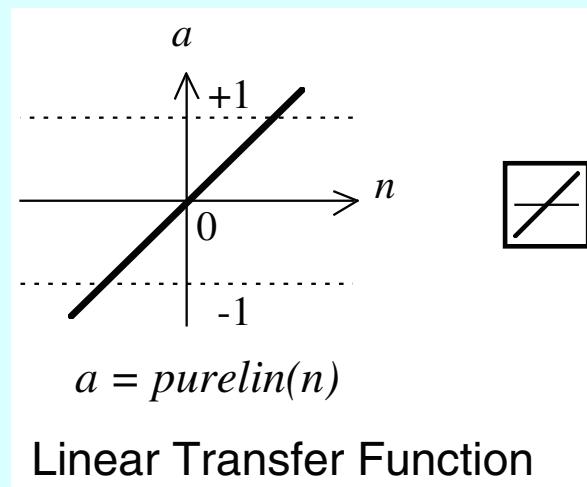
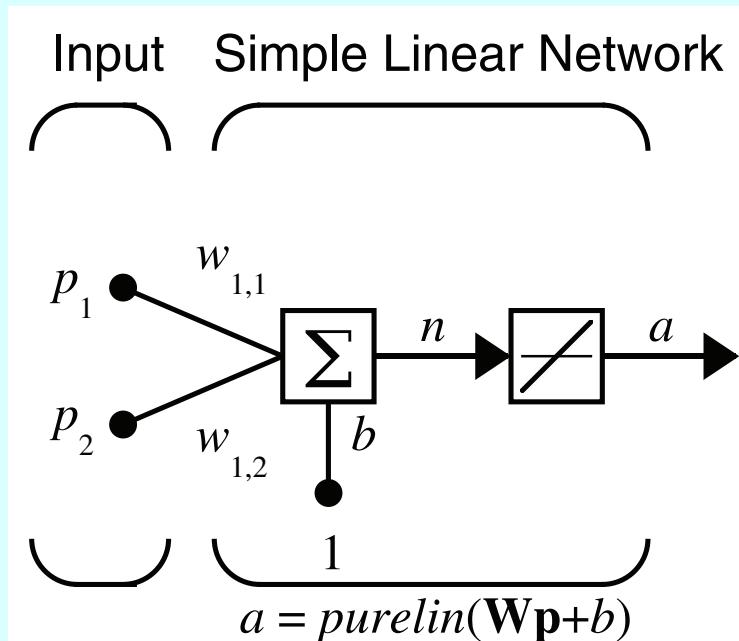
Здесь вектор w_i^T сформирован из элементов i -го столбца матрицы весов W :

$$w_i = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

Обучение сетей — правило Уидроу-Хоффа (VIII)

Сеть ADALINE – 7

Простейшая линейная сеть (1)

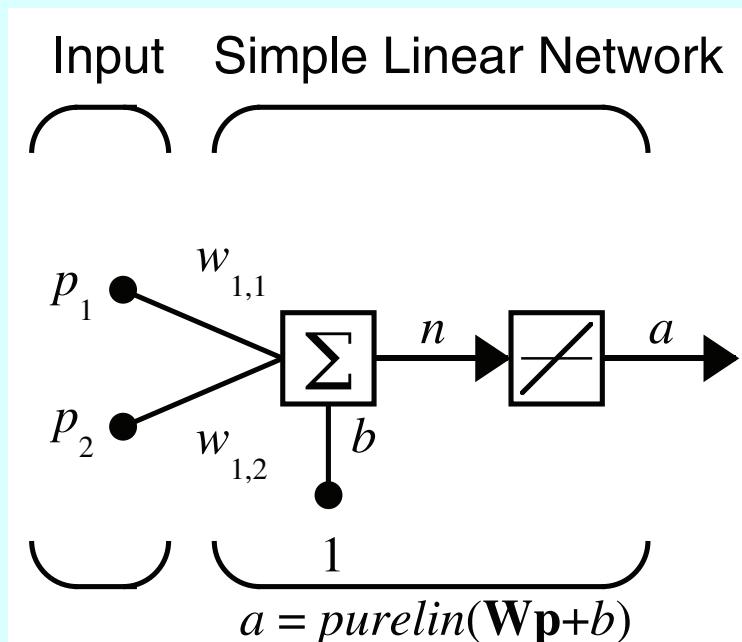


Источник: *Demuth H., Beale M., Hagan M.*. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (pp. 4-3, 4-5).

Обучение сетей — правило Уидроу-Хоффа (IX)

Сеть ADALINE – 8

Простейшая линейная сеть (2)



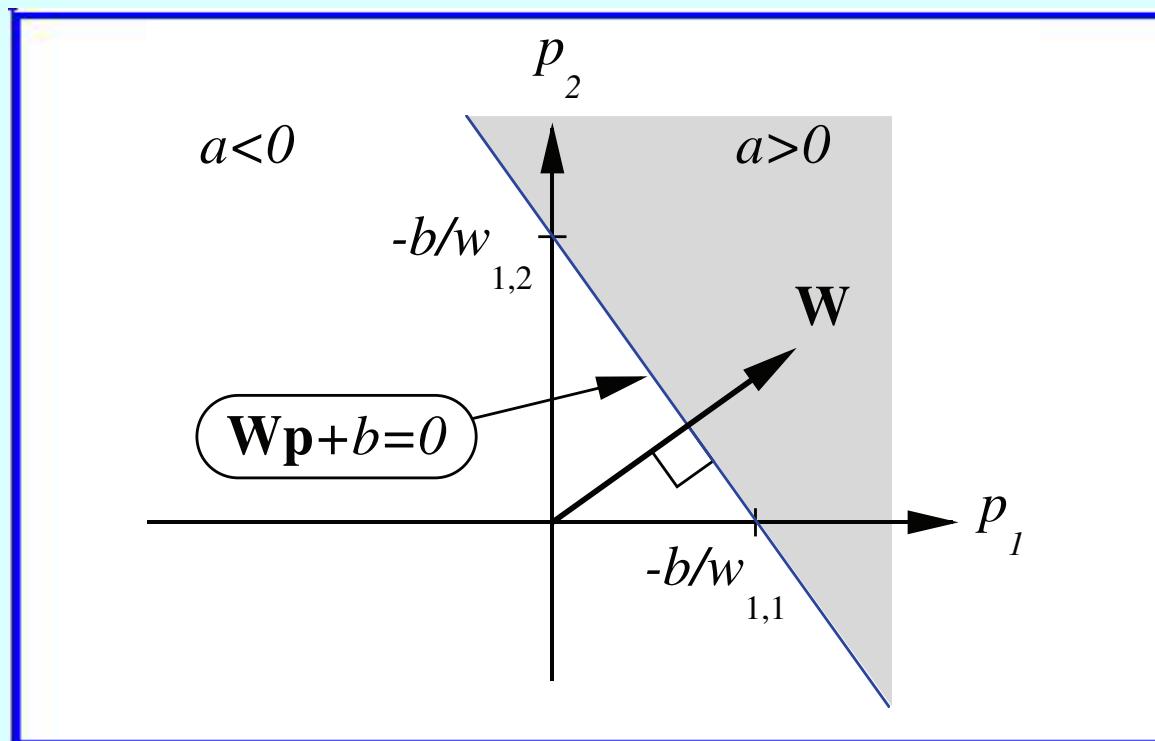
Выход сети:

$$\begin{aligned} a &= \text{purelin}(n) = \\ &= \text{purelin}(w_1^T p + b) = w_1^T p + b = \\ &= w_{1,1} p_1 + w_{1,2} p_2 + b \end{aligned}$$

Обучение сетей — правило Уидроу-Хоффа (X)

Сеть ADALINE – 9

Разделяющая граница



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 4-5).

Обучение сетей – правило Уидроу-Хоффа (XI)

Сеть ADALINE – 10

Алгоритм LMS (1)

LMS (Least Mean Square) — алгоритм **обучения с учителем**, предназначенный для обучения сети ADALINE на заданном обучающем наборе.

Обучающий набор и выходы сети:

$$\{\langle p^1, t^1 \rangle, \langle p^2, t^2 \rangle, \dots, \langle p^k, t^k \rangle, \dots, \langle p^L, t^L \rangle\}$$

p^k — k -й входной вектор, y^k — текущий выход сети; $p^k \rightarrow y^k$

t^k — эталонный (желаемый) выход, отвечающий вектору p^k

Общая идея алгоритма LMS:

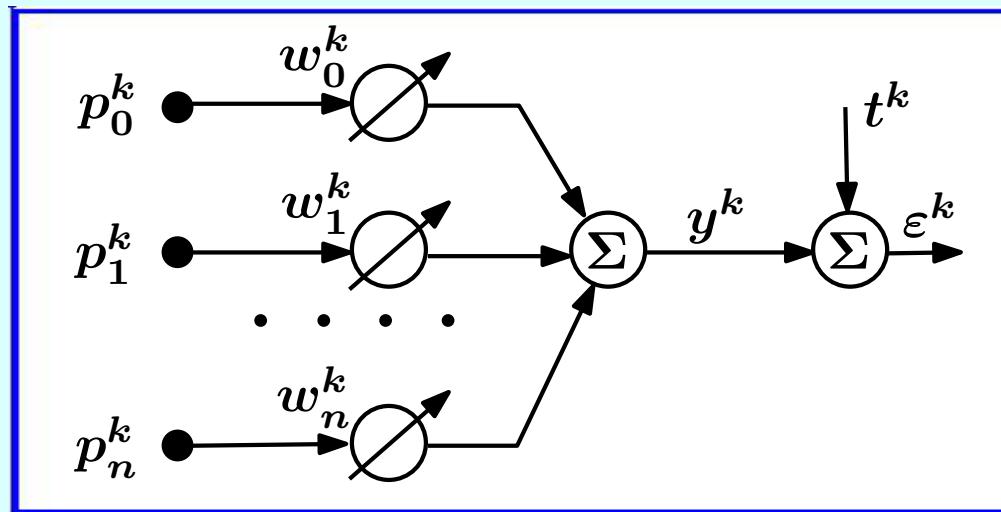
Алгоритм LMS подстраивает **веса и смещения** сети ADALINE таким образом, чтобы минимизировать **среднеквадратическую ошибку**, характеризующую уклонение **текущего выхода** сети (при данном текущем значении ее весов и смещений) и **целевого значения** этого выхода.

Обучение сетей – правило Уидроу-Хоффа (XII)

Сеть ADALINE – 11

Алгоритм LMS (2)

Вычисление ошибки для текущего входного вектора



$p_0^k = 1$, $w_0^k = b$ – смещение, t^k – требуемый (эталонный) выход,
 k – номер текущего входного вектора p^k , $k = 1, 2, \dots, L$

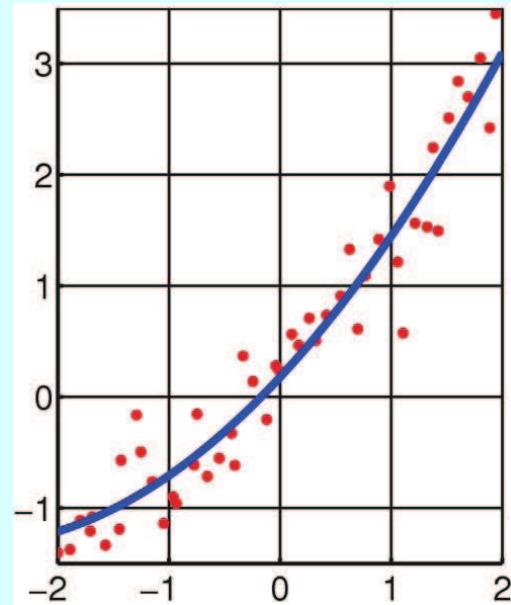
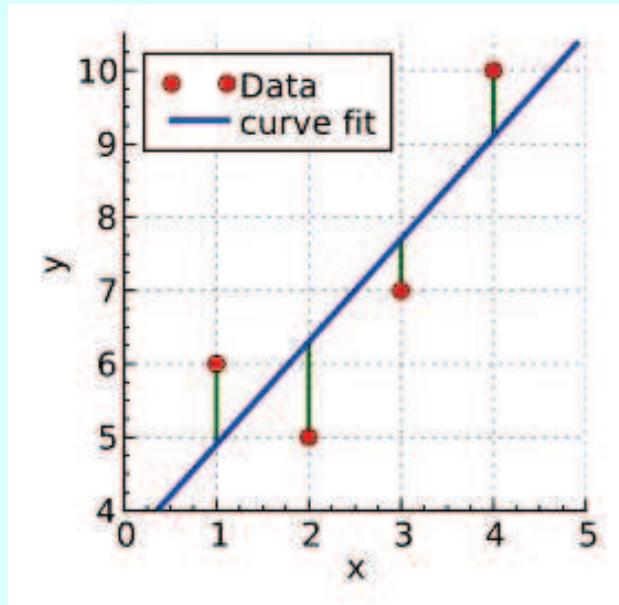
Источник: *Widrow B., Stearns S.D.* Adaptive signal processing. –
Prentice-Hall, Inc. – 1985. – 474 pp. (Figure 2.4, p. 19).

Обучение сетей — правило Уидроу-Хоффа (XIII)

Сеть ADALINE – 12

Алгоритм LMS (3)

Метод наименьших квадратов



Карл Гаусс (Carl Friedrich Gauss) — идея метода (1794), разработка (1821–1823).

Адриен-Мари Лежандр (Adrien-Marie Legendre) — разработка и публикация (1805–1806)

$$r_i = y_i - f(x_i, \beta); \quad S = \sum_{i=1}^n r_i^2; \quad S^* = S(\beta^*) = \min_{\beta} S(\beta)$$

Источники: *Least squares* and *Linear least squares*. – From Wikipedia.

Обучение сетей — правило Уидроу-Хоффа (XIV)

Сеть ADALINE – 13

Алгоритм LMS (4)

Выход сети для входного вектора p^k :

$$y^k = \sum_{j=0}^n w_j^k p_j^k = \sum_{j=1}^n w_j^k p_j^k + b^k$$

Ошибка сети для обучающего примера $\langle p^k, t^k \rangle$:

$$E^k = t^k - y^k$$

Квадратичная ошибка сети для примера $\langle p^k, t^k \rangle$:

$$E = \frac{1}{2}(E^k)^2 = \frac{1}{2}(t^k - y^k)^2$$

Суть предложения Уидроу и Хоффа состояла в использовании в качестве **минимизируемого показателя обученности** сети квадратичной ошибки $(E^k)^2$ сети на k -м шаге.

Обучение сетей — правило Уидроу-Хоффа (XV)

Сеть ADALINE – 14

Алгоритм LMS (5)

Квадратичная ошибка сети для примера $\langle p^k, t^k \rangle$:

$$E = \frac{1}{2}(E^k)^2 = \frac{1}{2}(t^k - y^k)^2$$

Градиент функции ошибки:

$$\nabla E = \left[\frac{\partial E}{\partial w_1} \ \frac{\partial E}{\partial w_2} \ \dots \ \frac{\partial E}{\partial w_n} \ \frac{\partial E}{\partial b} \right]^T$$

Обучение сетей — правило Уидроу-Хоффа (XVI)

Сеть ADALINE – 14

Алгоритм LMS (5)

Ошибка сети для примера $\langle p^k, t^k \rangle$:

$$E(w) = (E^k)^2 = \frac{1}{2}(t^k - y^k)^2$$

Правило обучения Уидроу-Хоффа:

$$w_j^{k+1} = w_j^k - \alpha \frac{\partial E^k}{\partial w_j^k}$$

$$b^{k+1} = b^k - \alpha \frac{\partial E^k}{\partial b^k}$$

Здесь $j = 1, 2, \dots, n$, α — скорость обучения

Другое название (более часто используемое) для правила Уидроу-Хоффа — **дельта-правило**.

Правило Уидроу-Хоффа базируется на **градиентном спуске** в пространстве весов w и смещений b нейронной сети.

Обучение сетей — правило Уидроу-Хоффа (XVII)

Сеть ADALINE – 15

Алгоритм LMS (6)

Выход и ошибка сети для примера $\langle p^k, t^k \rangle$:

$$y^k = \sum_{j=1}^n w_j^k p_j^k + b^k, \quad E(w, b) = \frac{1}{2} (E^k)^2 = \frac{1}{2} (y^k - t^k)^2$$

Правило обучения Уидроу-Хоффа:

$$w_j^{k+1} = w_j^k - \alpha \frac{\partial E}{\partial w_j^k}, \quad b^{k+1} = b^k - \alpha \frac{\partial E}{\partial b^k}$$

Производные ошибки сети E по настраиваемым параметрам сети w_j и b :

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial y^k} \frac{\partial y^k}{\partial w_j} = -(t^k - y^k) p_j^k = -E^k p_j^k$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial y^k} \frac{\partial y^k}{\partial b} = -(t^k - y^k) = -E^k$$

Обучение сетей – правило Уидроу-Хоффа (XVIII)

Сеть ADALINE – 16

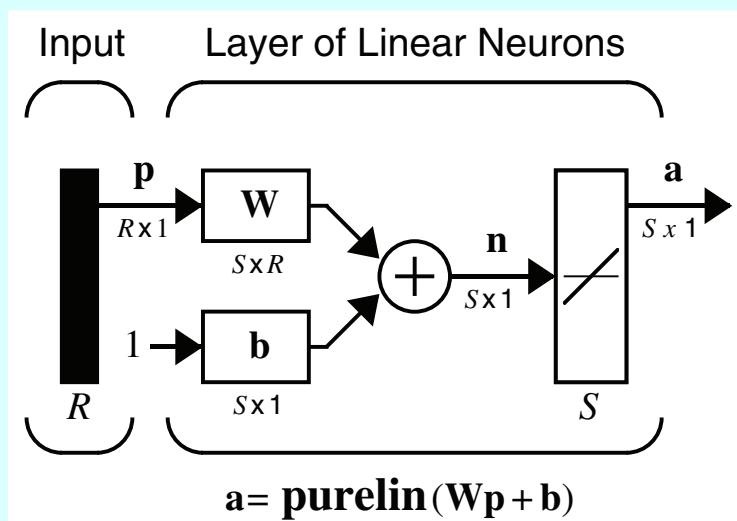
Алгоритм LMS (7)

Матричная форма дельта-правила
для ADALINE:

$$\mathbf{W}^{k+1} = \mathbf{W}^k + \alpha \mathbf{E}^k (\mathbf{p}^k)^T$$

$$\mathbf{b}^{k+1} = \mathbf{b}^k + \alpha \mathbf{E}^k$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_S^T \end{bmatrix}, \quad \mathbf{w}_i = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$



Здесь \mathbf{E}^k и \mathbf{b}^k – векторы размерности S ,
 \mathbf{W} – матрица размерности $(S \times R)$, $w_{i,j}$ – вес связи между i -м элементом сети ($i = 1, 2, \dots, S$) и j -й компонентой входного вектора ($j = 1, 2, \dots, R$).

Условие сходимости алгоритма LMS: $0 < \alpha < 1/\lambda_{max}$, где λ_{max} – наибольшее собственное значение корреляционной матрицы входов \mathbf{R} .

Обучение сетей — правило Уидроу-Хоффа (XIX)

Сеть ADALINE – 17

Алгоритм LMS (8)

Процедура обучения сети ADALINE:

1. Задаются **скорость обучения** α , ($0 < \alpha < 1$) и **минимальная ошибка** сети E_{min} , которой необходимо достичь в процессе обучения.
2. Весовые коэффициенты W и смещения b сети инициализируются **случайным образом**.
3. На входы сети поочередно подаются **входные векторы** p^k из обучающего набора, преобразуемые в **выходные сигналы** a^k .
4. Весовые коэффициенты W и смещения b сети **корректируются** согласно дельта-правилу

$$W^{k+1} = W^k + \alpha E^k (p^k)^T$$

$$b^{k+1} = b^k + \alpha E^k$$

5. Процедура выполняется до тех пор, пока ошибка сети не станет меньше заданной, т.е. $E \leq E_{min}$.

Обучение сетей — правило Уидроу-Хоффа (XX)

Сеть ADALINE – 18

Пример обучения сети ADALINE (1)

Пример: Сортировка яблок и апельсинов

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = -1 \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = 1 \right\}$$

Дельта-правило для ADALINE без смещений:

$$\mathbf{W}^{k+1} = \mathbf{W}^k + \alpha \mathbf{E}^k (\mathbf{p}^k)^T$$

Корреляционная матрица входов сети:

$$\begin{aligned} \mathbf{R} &= \frac{1}{2} \mathbf{p}_1 \mathbf{p}_1^T + \frac{1}{2} \mathbf{p}_2 \mathbf{p}_2^T = \\ &= \frac{1}{2} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} [1 \ -1 \ -1] + \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} [1 \ 1 \ -1] = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \end{aligned}$$

Собственные значения матрицы R и скорость обучения α :

$$\lambda_1 = 1.0, \quad \lambda_2 = 0.0, \quad \lambda_3 = 2.0; \quad \alpha < \frac{1}{\lambda_{max}} = \frac{1}{2.0} = 0.5$$

Обучение сетей — правило Уидроу-Хоффа (XXI)

Сеть ADALINE – 19

Пример обучения сети ADALINE (2)

Шаг 1

На входе вектор p_1 (апельсин),
начальные значения весов — нулевые, $\alpha = 0.2$:

$$a(0) = W(0)p(0) = W(0)p_1 = [0 \ 0 \ 0] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = 0$$

$$E(0) = t(0) - a(0) = t_1 - a(0) = -1 - 0 = -1$$

Новое значение весовой матрицы W :

$$\begin{aligned} W(1) &= W(0) + 2\alpha E(0)p^T(0) = \\ &= [0 \ 0 \ 0] + 2(0.2)(-1) \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}^T = [-0.4 \ 0.4 \ 0.4] \end{aligned}$$

Обучение сетей — правило Уидроу-Хоффа (XXII)

Сеть ADALINE – 20

Пример обучения сети ADALINE (3)

Шаг 2

На входе вектор p_2 (яблоко):

$$a(1) = \mathbf{W}(1)\mathbf{p}(1) = \mathbf{W}(1)p_2 = [-0.4 \ 0.4 \ 0.4] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = -0.4$$

$$E(1) = t(1) - a(1) = t_2 - a(1) = -1 - (-0.4) = 1.4$$

Новое значение весовой матрицы \mathbf{W} :

$$\begin{aligned} \mathbf{W}(2) &= \mathbf{W}(1) + 2\alpha E(1) \mathbf{p}^T(1) = \\ &= [-0.4 \ 0.4 \ 0.4] + 2(0.2)(1.4) \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}^T = [0.16 \ 0.96 \ -0.16] \end{aligned}$$

Обучение сетей – правило Уидроу-Хоффа (XXIII)

Сеть ADALINE – 21

Пример обучения сети ADALINE (4)

Шаг 3

На входе опять вектор p_1 (апельсин):

$$a(2) = \mathbf{W}(2)\mathbf{p}(2) = \mathbf{W}(2)p_1 = [0.16 \ 0.96 \ -0.16] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = -0.64$$

$$E(2) = t(2) - a(2) = t_1 - a(2) = -1 - (-0.64) = -0.36$$

Новое значение весовой матрицы \mathbf{W} :

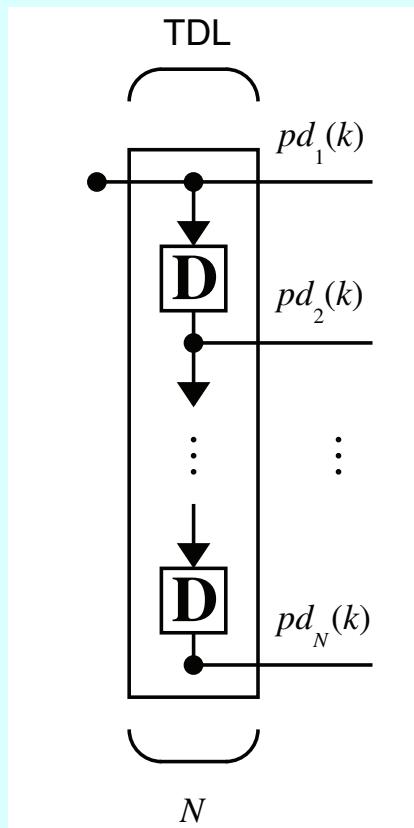
$$\begin{aligned} \mathbf{W}(3) &= \mathbf{W}(2) + 2\alpha E(2) \mathbf{p}^T(2) = \\ &= [0.16 \ 0.96 \ -0.16] + 2(0.2)(-0.36) \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}^T = [0.016 \ 1.1040 \ -0.016] \end{aligned}$$

При числе шагов $N \rightarrow \infty$ алгоритм сходится к $\mathbf{W}(\infty) = [0 \ 1 \ 0]$.

Обучение сетей — правило Уидроу-Хоффа (XXIV)

Сеть ADALINE – 22

Адаптивная фильтрация (1)



Линия задержки с отводами

TDL — Tapped Delay Line

TDL — Time-Delay Line

Вход TDL — отсчет $y(k)$ входного сигнала $y(t)$ в момент времени $t = k$

Выход TDL — N -мерный вектор:

$$pd_1(k) = y(k)$$

$$pd_2(k) = y(k - 1)$$

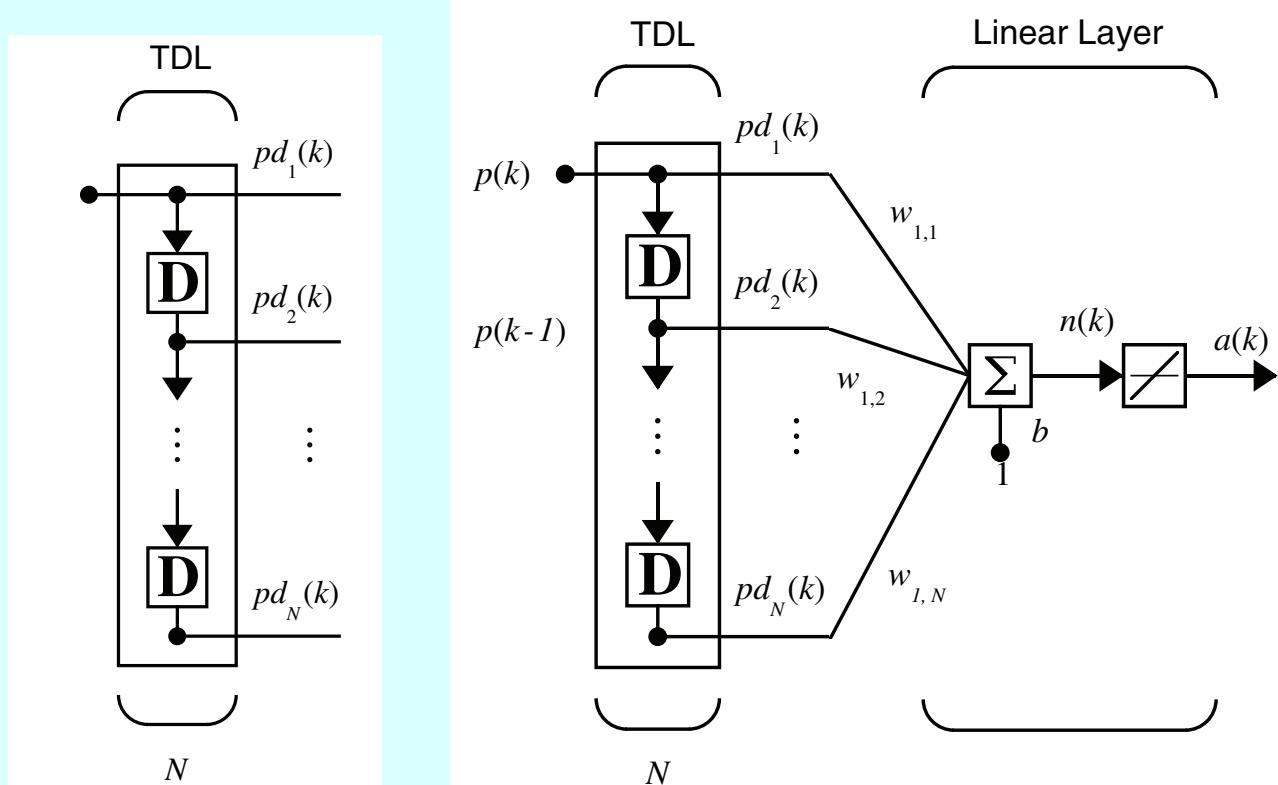
...

$$pd_N(k) = y(k - N + 1)$$

ADALINE в сочетании с TDL широко используется в адаптивной фильтрации.

Обучение сетей – правило Уидроу-Хоффа (XXV)

Сеть ADALINE – 23 Адаптивная фильтрация (2)



$$pd_1(k) = \mathbf{y}_k, \quad pd_2(k) = \mathbf{y}_{k-1}, \dots, \quad pd_N(k) = \mathbf{y}_{k-N+1}$$

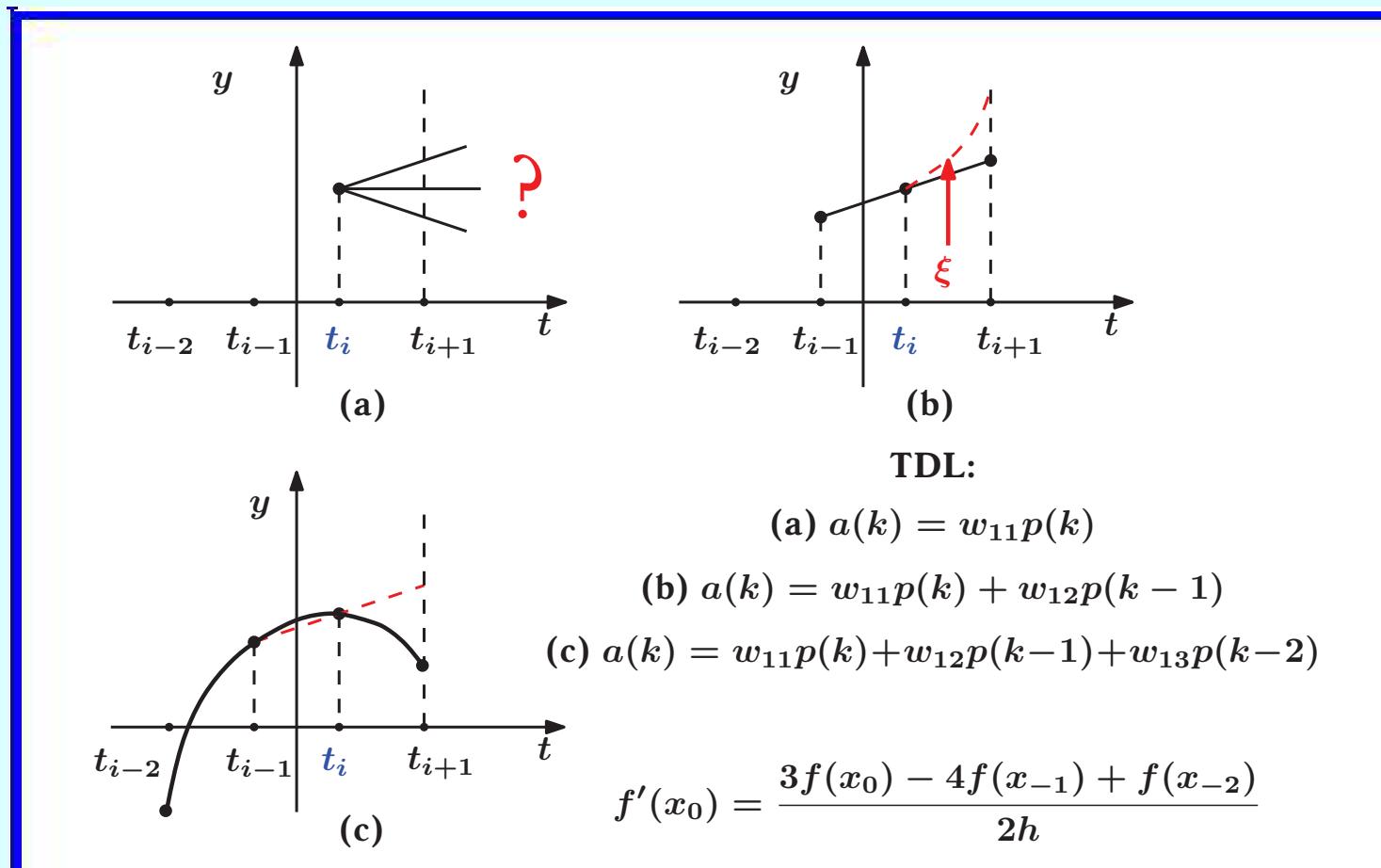
$$a(k) = \text{purelin}(\mathbf{W}\mathbf{p} + b) = \sum_{i=1}^N w_{1,i} \mathbf{y}(k-i+1) + b$$

Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 10, Figs. 10-4 and 10-5, p.10-14).

Обучение сетей – правило Уидроу-Хоффа (XXVI)

Сеть ADALINE – 24

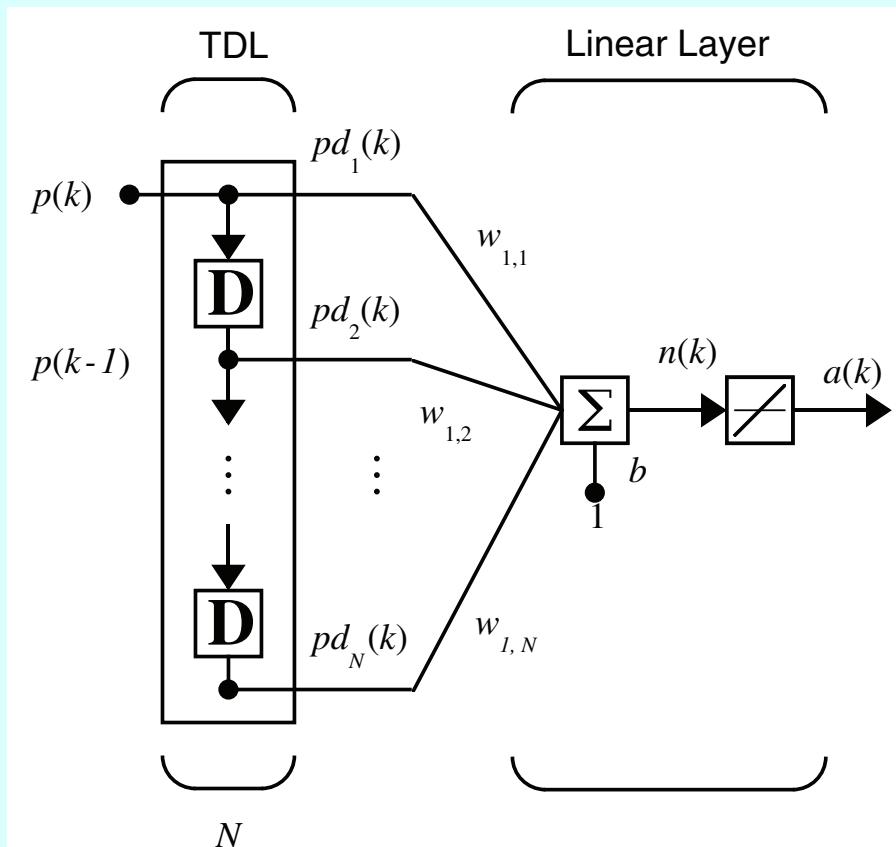
Адаптивная фильтрация (3)
Предыстория процесса



Обучение сетей — правило Уидроу-Хоффа (XXVII)

Сеть ADALINE – 25

Адаптивная фильтрация (4)



FIR (Finite Impulse Response) — фильтр с конечной импульсной характеристикой

Вход FIR — отсчеты $y(k)$ входного сигнала $y(t)$ в моменты времени $t = k$, $k = 1, 2, \dots$

Выход TDL — N -мерный вектор:
 $pd_1(k) = y(k)$

$$pd_2(k) = y(k - 1)$$
$$\dots$$

$$pd_N(k) = y(k - N + 1)$$

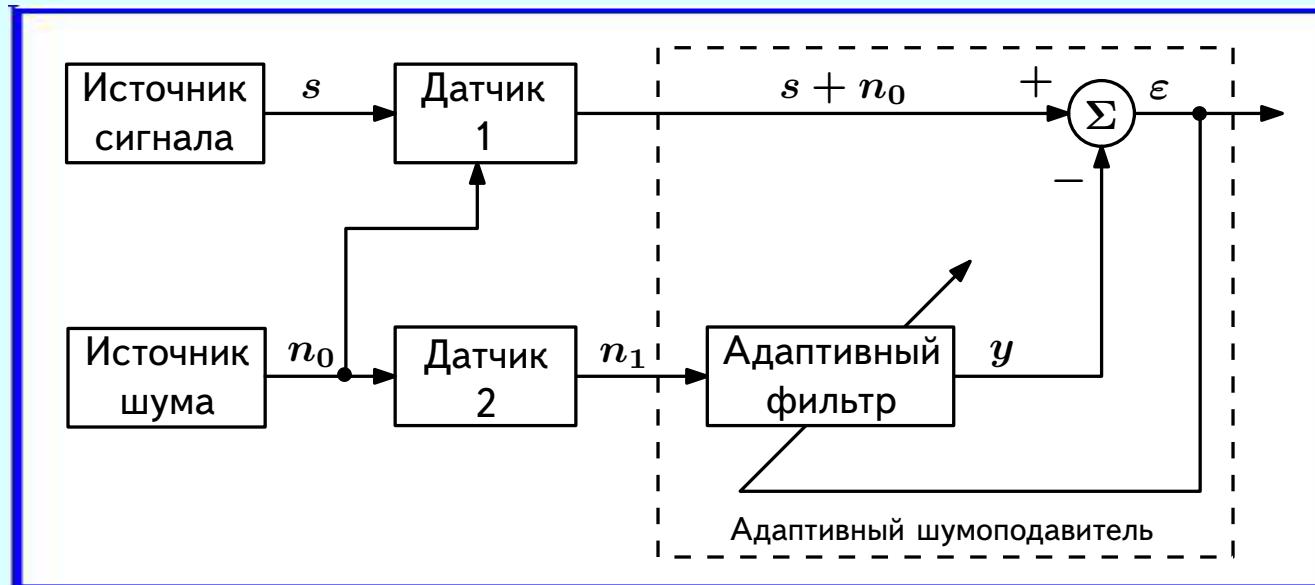
Выход FIR:

$$a(k) = \text{purelin}(Wp + b) =$$
$$= \sum_{i=1}^N w_{1,i} y(k - i + 1) + b$$

Обучение сетей — правило Уидроу-Хоффа (XXVIII)

Сеть ADALINE – 26

Адаптивное шумоподавление (1)



s — «чистый» входной сигнал; n_0 — помеха, не коррелированная с s ;
 $(s + n_0)$ — входной сигнал шумоподавителя (смесь сигнала и помехи);

n_1 — помеха, не коррелированная с s ,

но некоторым неизвестным образом коррелированная с n_0 ;

y — выходной сигнал, полученный фильтрацией помехи n_1 ,
приблизительная копия помехи n_0 ;

ϵ — выход шумоподавителя

Обучение сетей – правило Уидроу-Хоффа (XXIX)

Сеть ADALINE – 27

Адаптивное шумоподавление (2)

Пусть s , n_0 , n_1 и y – стационарные случайные процессы с нулевыми средними значениями, сигнал s не коррелирован с n_0 и n_1 , сигналы n_0 и n_1 – коррелированы.

Выходой сигнал адаптивного шумоподавителя:

$$\epsilon = s + n_0 - y$$

$$\epsilon^2 = (s + n_0 - y)^2 = s^2 + (n_0 - y)^2 + 2s(n_0 - y)$$

Математическое ожидание квадрата выходного сигнала:

$$E[\epsilon^2] = E[s^2] + E[(n_0 - y)^2] + 2E[s(n_0 - y)] = E[s^2] + E[(n_0 - y)^2]$$

Цель настройки адаптивного фильтра:

$$E_{min}[\epsilon^2] = \min_W E[\epsilon^2]$$

Мощность сигнала $E[s^2]$ не изменяется при перестройке фильтра в процессе минимизации $E[\epsilon^2]$ (используя, например, LMS), поэтому

$$E_{min}[\epsilon^2] = E[s^2] + E_{min}[(n_0 - y)^2]$$

Обучение сетей — правило Уидроу-Хоффа (XXX)

Сеть ADALINE – 28

Адаптивное шумоподавление (3)

В процессе минимизации $E[\varepsilon^2]$ мощность сигнала $E[s^2]$ не изменяется:

$$E_{min}[\varepsilon^2] = E[s^2] + E_{min}[(n_0 - y)^2]$$

Если $E[\varepsilon^2]$ минимально, то минимально также и $E_{min}[(n_0 - y)^2]$.

Тогда **выходной сигнал фильтра y** является **наилучшей среднеквадратической оценкой** помехи n_0 .

При минимальном значении $E_{min}[(n_0 - y)^2]$ **минимальное значение** имеет также и $E_{min}[(\varepsilon - s)^2]$, поскольку

$$\varepsilon = s + n_0 - y \Rightarrow (\varepsilon - s) = (n_0 - y)$$

Наименьшая возможная мощность выходного сигнала

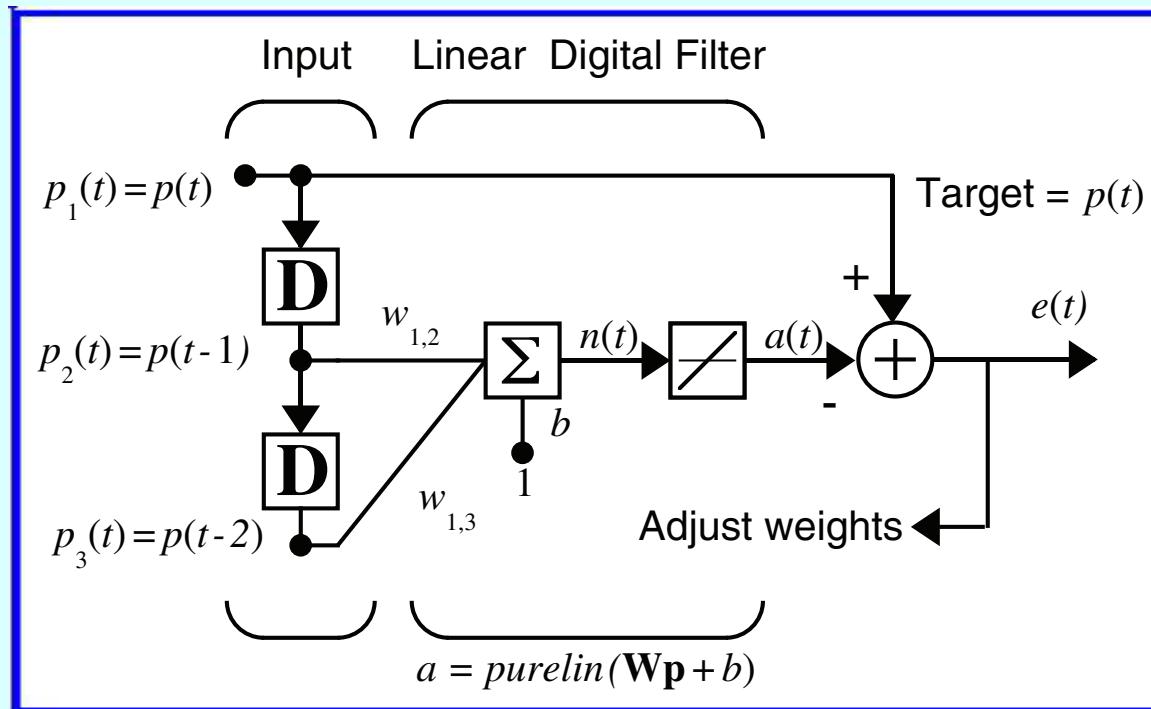
$$E_{min}[\varepsilon^2] = E[s^2] \Rightarrow E[(n_0 - y)^2] = 0 \Rightarrow y = n_0 \Rightarrow \varepsilon = s$$

Таким образом, минимизация общей мощности выходного сигнала $E[\varepsilon^2]$ в данной схеме приводит к получению **наилучшего в среднеквадратическом смысле приближения** сигнала s .

Обучение сетей – правило Уидроу-Хоффа (XXXI)

Сеть ADALINE – 29

Адаптивный экстраполатор (1)



Адаптивный экстраполатор (adaptive predictor): предсказать **следующее значение** стационарного случайного процесса $p(t)$ по двум его значениям в предшествующие моменты времени ($p(t - 1)$, $p(t - 2)$).

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 10-13).

Обучение сетей — правило Уидроу-Хоффа (XXXII)

Сеть ADALINE – 30

Адаптивный экстраполятор (2)

Пусть дан временной ряд $p(t)$ на промежутке $t = 1, 2, \dots, n, \dots, m - 1, m$:

$$p(1), p(2), \dots, p(n), \dots, p(m - 1), p(m)$$

Найти $p(m + 1), p(m + 2), \dots$

Обучающая выборка как матрица \mathbf{X} :

$$\mathbf{X} = \begin{bmatrix} p(1) & p(2) & \dots & p(q) \\ p(2) & p(3) & \dots & p(q + 1) \\ \dots & \dots & \dots & \dots \\ p(m - q) & p(m - q + 1) & \dots & p(m - 1) \end{bmatrix}$$

Здесь m — размерность обучающей выборки; q — длина скользящего окна.

В сети будет q распределительных (входных) нейронов и один выходной нейрон.

Обучение сетей — правило Уидроу-Хоффа (XXXIII)

Сеть ADALINE – 31

Адаптивный экстраполятор (3)

Адаптивный экстраполятор реализует модель **линейной авторегрессии**:

$$\hat{p}(n) = \sum_{k=1}^q w_k p(n - q + k - 1)$$

Здесь n — текущий момент времени; $\hat{p}(n)$ — оценка значения ряда $p(n)$ в момент времени n ; w_k — синаптические **веса** сети; q — длина **скользящего окна**.

Ошибка прогнозирования:

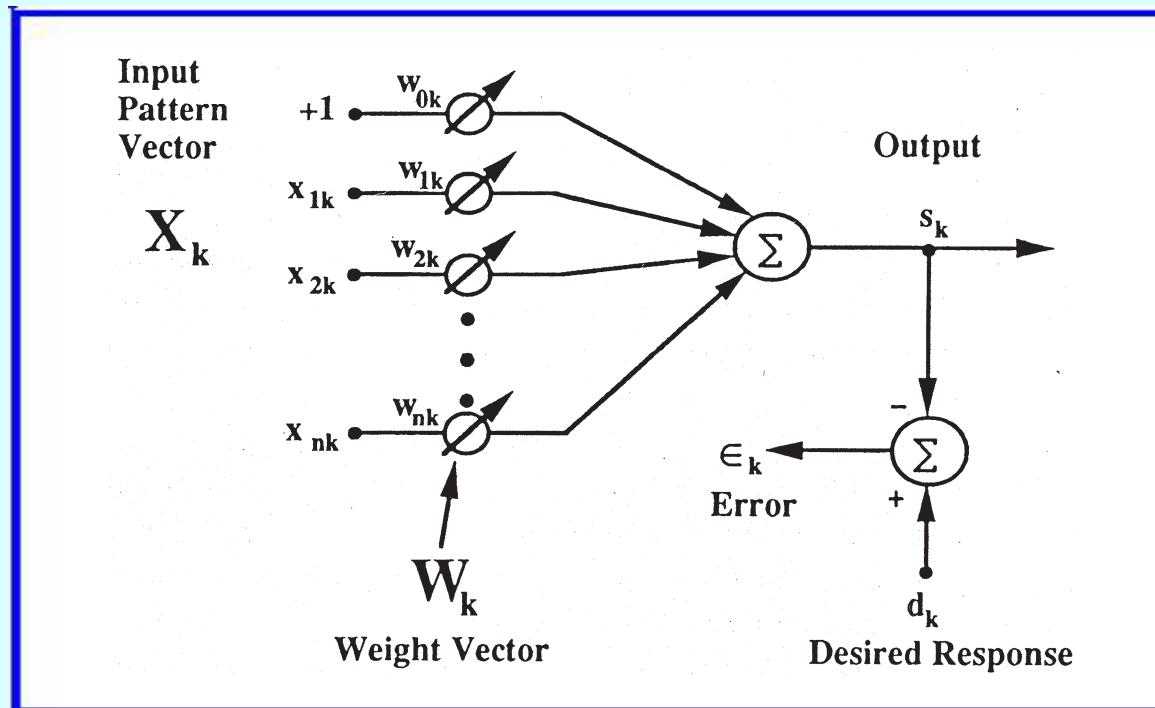
$$\varepsilon(n) = p(n) - \hat{p}(n)$$

Модель линейной авторегрессии формирует значения ряда $p(n)$ как **взвешенную сумму предыдущих значений** ряда.

Обучение сетей — правило Уидроу-Хоффа (XXXIV)

Сеть ADALINE – 32

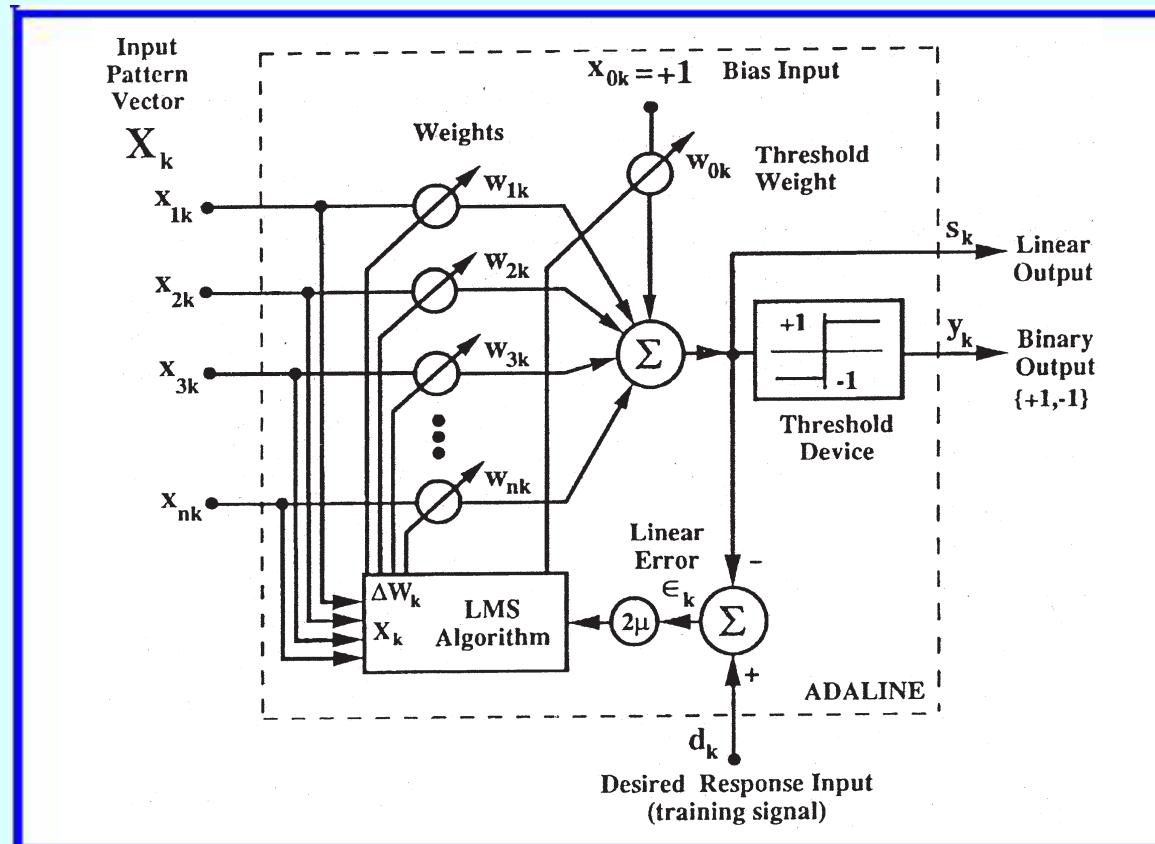
Adaptive Linear Combiner



Источник: Widrow B., Lehr M.A. Artificial neural networks of the Perceptron, Madaline, and backpropagation family // In: “Neurobionics” / H.-W. Bothe, M. Samii and R. Eckmiller (Editors). – Horth-Holland, 1993. – pp. 133–205 (Figure 1, p. 139).

Обучение сетей — правило Уидроу-Хоффа (XXXV)

Сеть ADALINE – 33 ADALINE — вариант Б. Уидроу

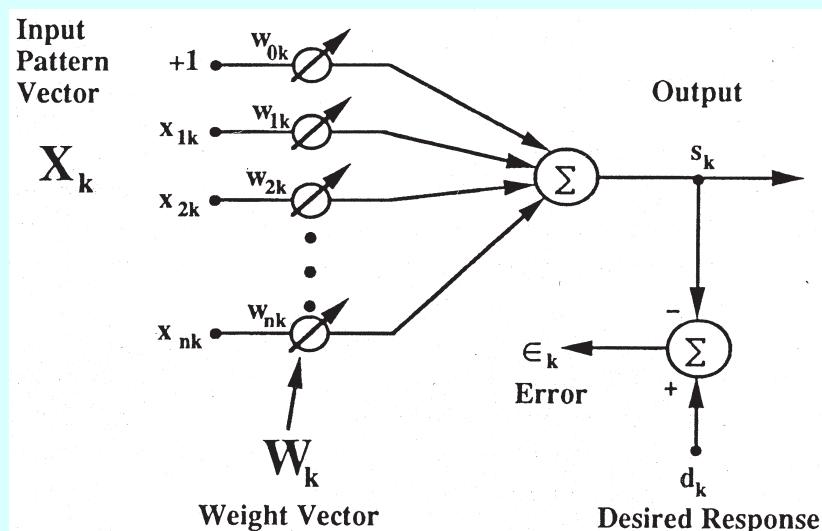


Источник: Widrow B., Lehr M.A. Artificial neural networks of the Perceptron, Madaline, and backpropagation family // In: "Neurobionics" / H.-W. Bothe, M. Samii and R. Eckmiller (Editors). – Horth-Holland, 1993. – pp. 133–205 (Figure 2, p. 141).

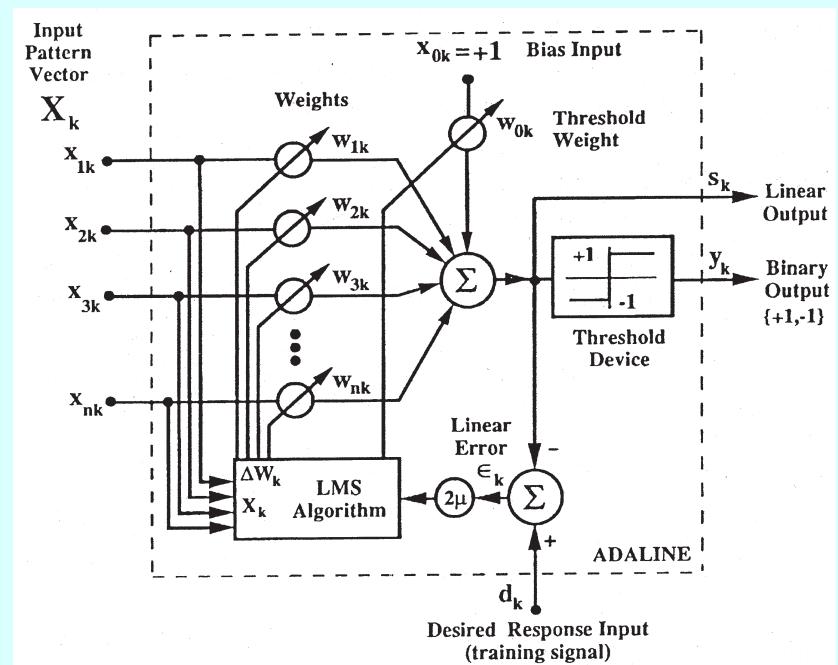
Обучение сетей — правило Уидроу-Хоффа (XXXVI)

Сеть ADALINE – 34

Adaptive Linear Combiner VS ADALINE



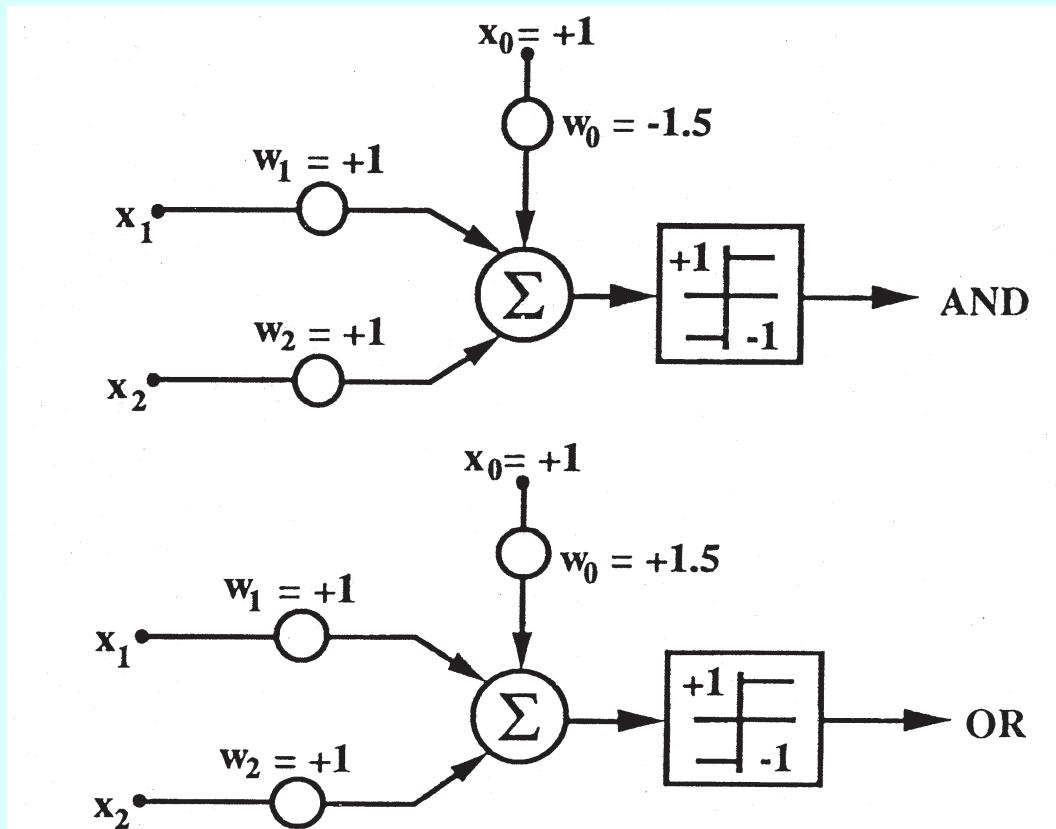
Adaptive Linear Combiner



ADALINE — вариант Б. Уидроу

Обучение сетей — правило Уидроу-Хоффа (XXXVII)

Сеть ADALINE – 35



Реализация логических функций
с помощью сети ADALINE
(a , y — выходы сумматора и сети)

$$a < 0 \Rightarrow y = -1$$

$$a \geq 0 \Rightarrow y = +1$$

AND:

$$(0, 0), (0, 1), (1, 0) \Rightarrow -1$$

$$(1, 1) \Rightarrow +1$$

$$a = x_1 + x_2 - 1.5$$

OR:

$$(0, 0) \Rightarrow -1$$

$$(0, 1), (1, 0), (1, 1) \Rightarrow +1$$

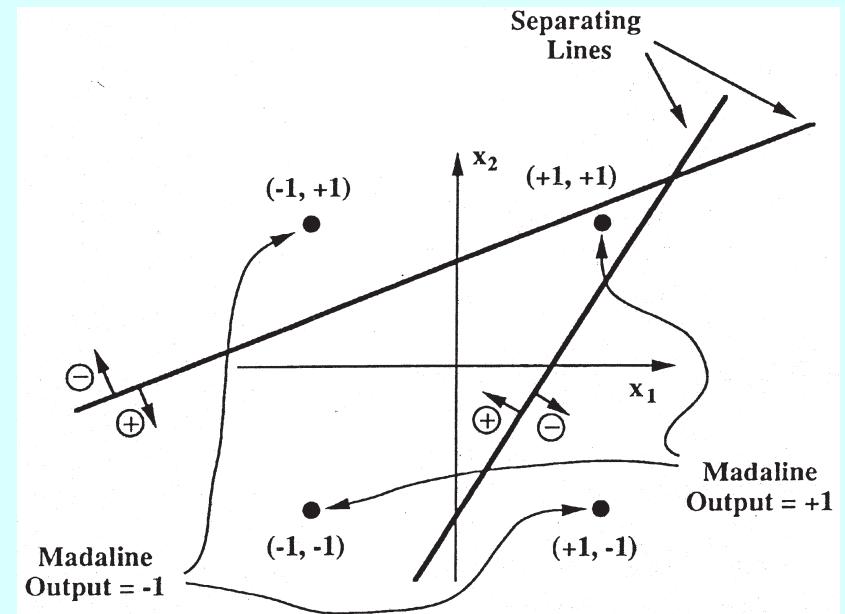
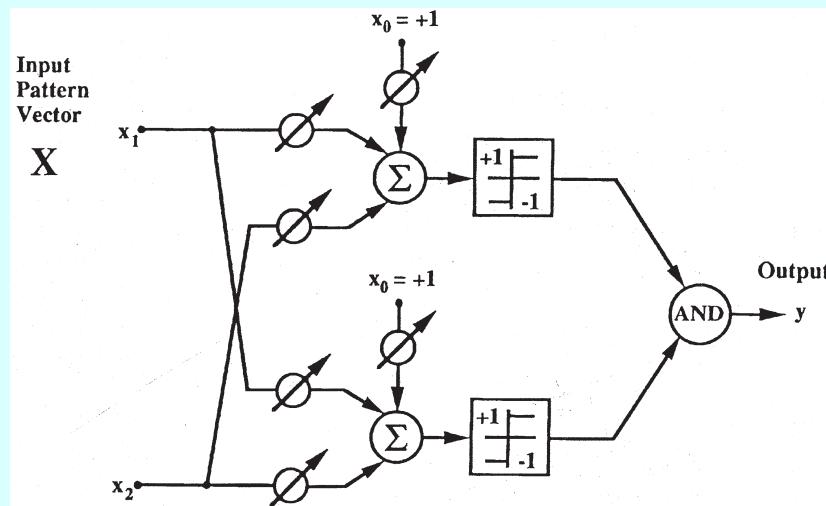
$$a = x_1 + x_2 + 1.5$$

Источник: Widrow B., Lehr M.A. Artificial neural networks of the Perceptron, Madaline, and backpropagation family // In: “Neurobionics” / H.-W. Bothe, M. Samii and R. Eckmiller (Editors). – Horth-Holland, 1993. – pp. 133–205 (Figure 7, p. 146).

Обучение сетей — правило Уидроу-Хоффа (XXXVIII)

Сеть ADALINE – 36

MADALINE — комбинация сетей ADALINE



MADALINE из двух ADALINE

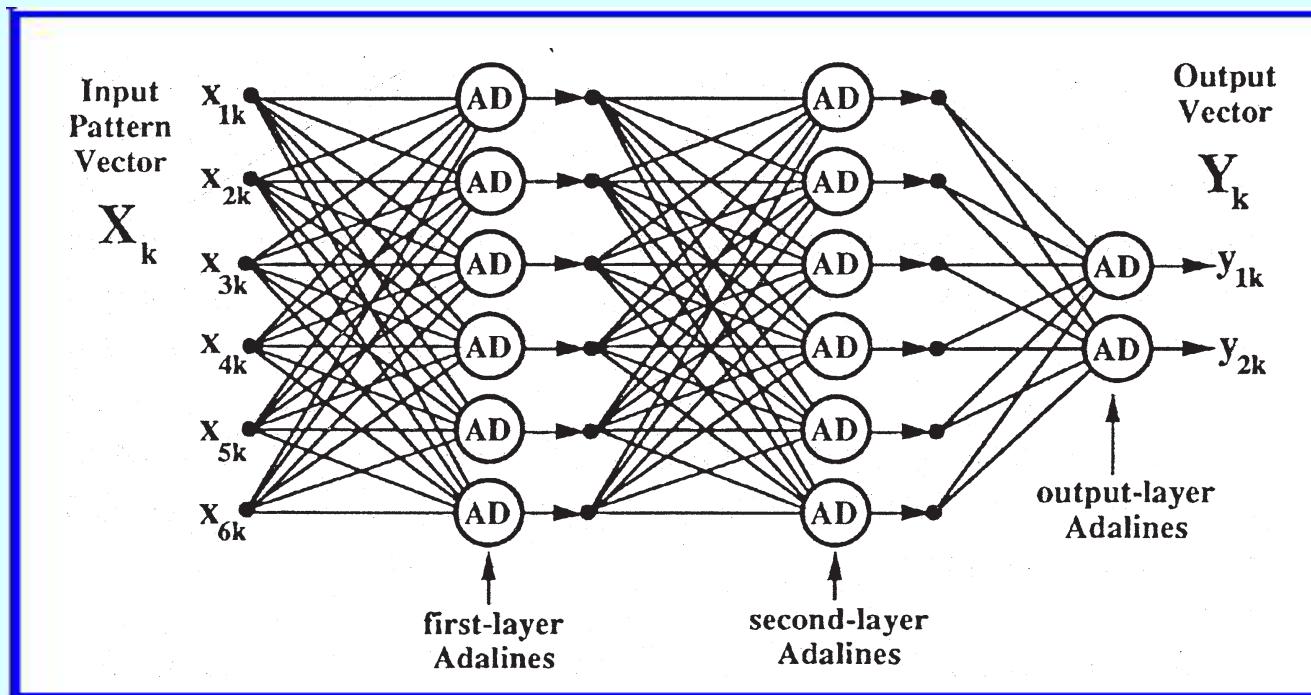
Разделяющие границы MADALINE

Источник: Widrow B., Lehr M.A. Artificial neural networks of the Perceptron, Madaline, and backpropagation family // In: “Neurobionics” / H.-W. Bothe, M. Samii and R. Eckmiller (Editors). – Horth-Holland, 1993. – pp. 133–205 (Figs 5, 6, p. 145).

Обучение сетей — правило Уидроу-Хоффа (XXXIX)

Сеть ADALINE – 37

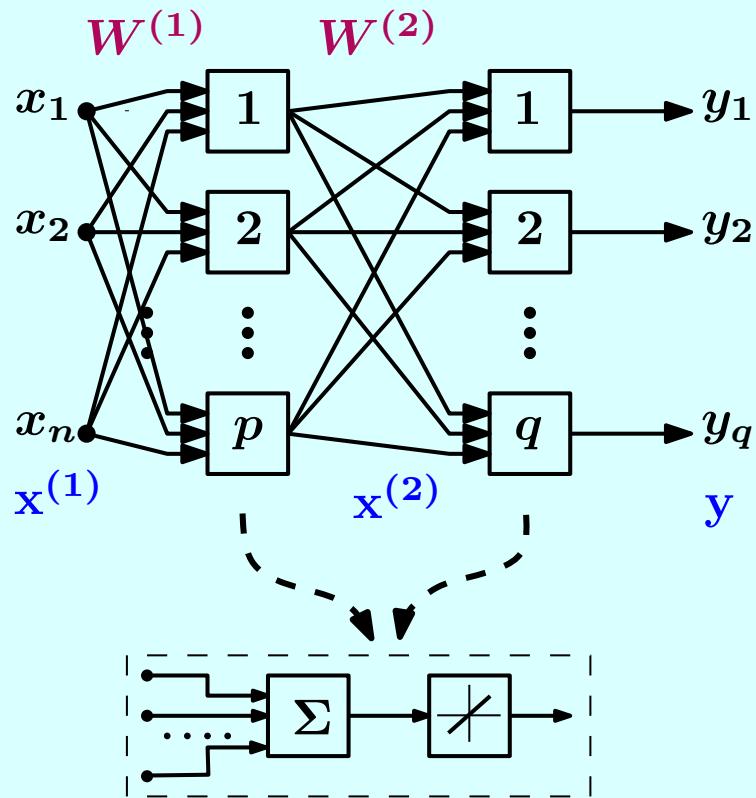
Многослойная сеть на основе ADALINE



Источник: Widrow B., Lehr M.A. Artificial neural networks of the Perceptron, Madaline, and backpropagation family // In: “Neurobionics” / H.-W. Bothe, M. Samii and R. Eckmiller (Editors). – Horth-Holland, 1993. – pp. 133–205 (Figure 8, p. 147).

Многослойные сети прямого распространения (I)

Линейные многослойные сети



$$x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})^T$$

$$x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \dots, x_p^{(2)})^T$$

$$y = (y_1, y_2, \dots, y_q)^T$$

$$W^{(1)} = \|w_{ij}^{(1)}\|, \quad i = 1, \dots, n; \quad j = 1, \dots, p$$

$$W^{(2)} = \|w_{jk}^{(2)}\|, \quad j = 1, \dots, p; \quad k = 1, \dots, q$$

$$y = x^{(2)} W^{(2)}$$

$$x^{(2)} = x^{(1)} W^{(1)}$$

$$y = x^{(1)} W^{(1)} W^{(2)}$$

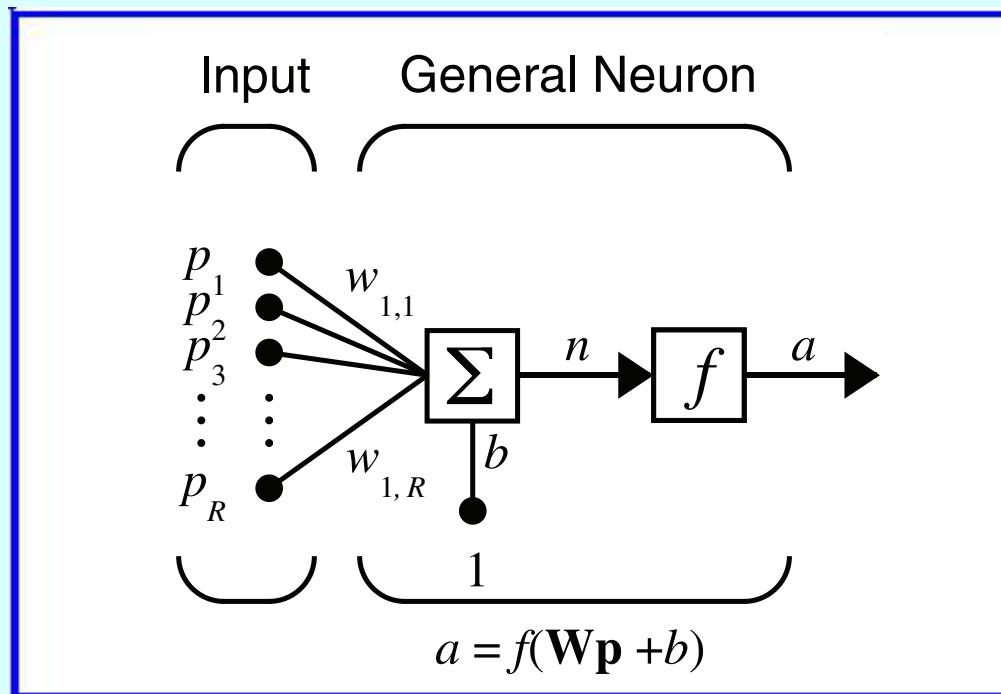
$$W = W^{(1)} W^{(2)}$$

$$y = x^{(1)} W$$

Линейная многослойная сеть с входами $x^{(1)}$ и матрицами весов межслойных связей $W^{(s)}$, $s = 1, \dots, N$ **эквивалентна однослойной сети** с теми же входами и матрицей весов $W = \prod_{s=1}^N W^{(s)}$.

Многослойные сети прямого распространения (II)

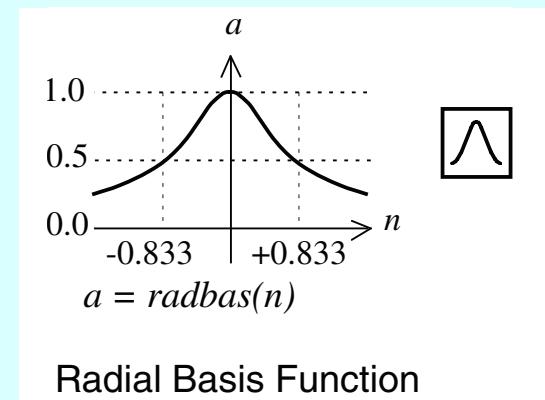
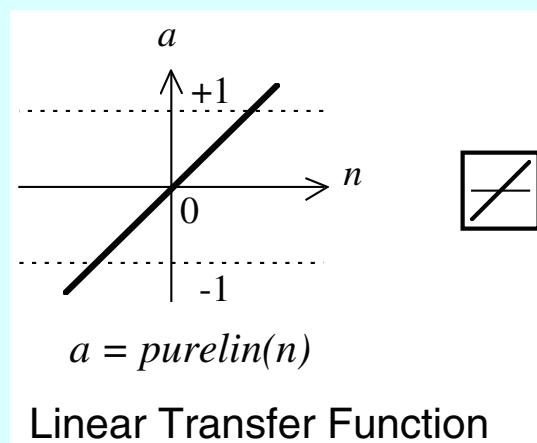
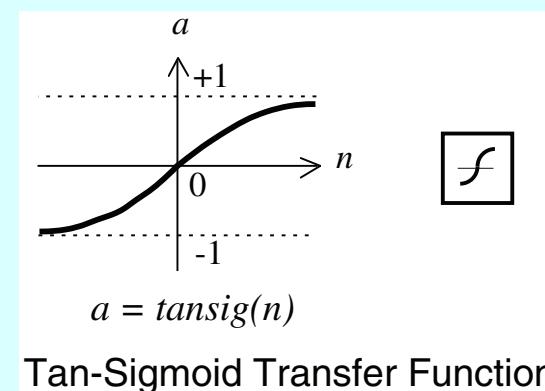
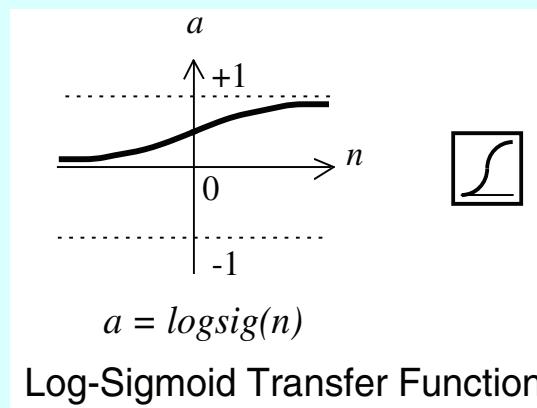
Нелинейный нейрон



Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-8).

Многослойные сети прямого распространения (III)

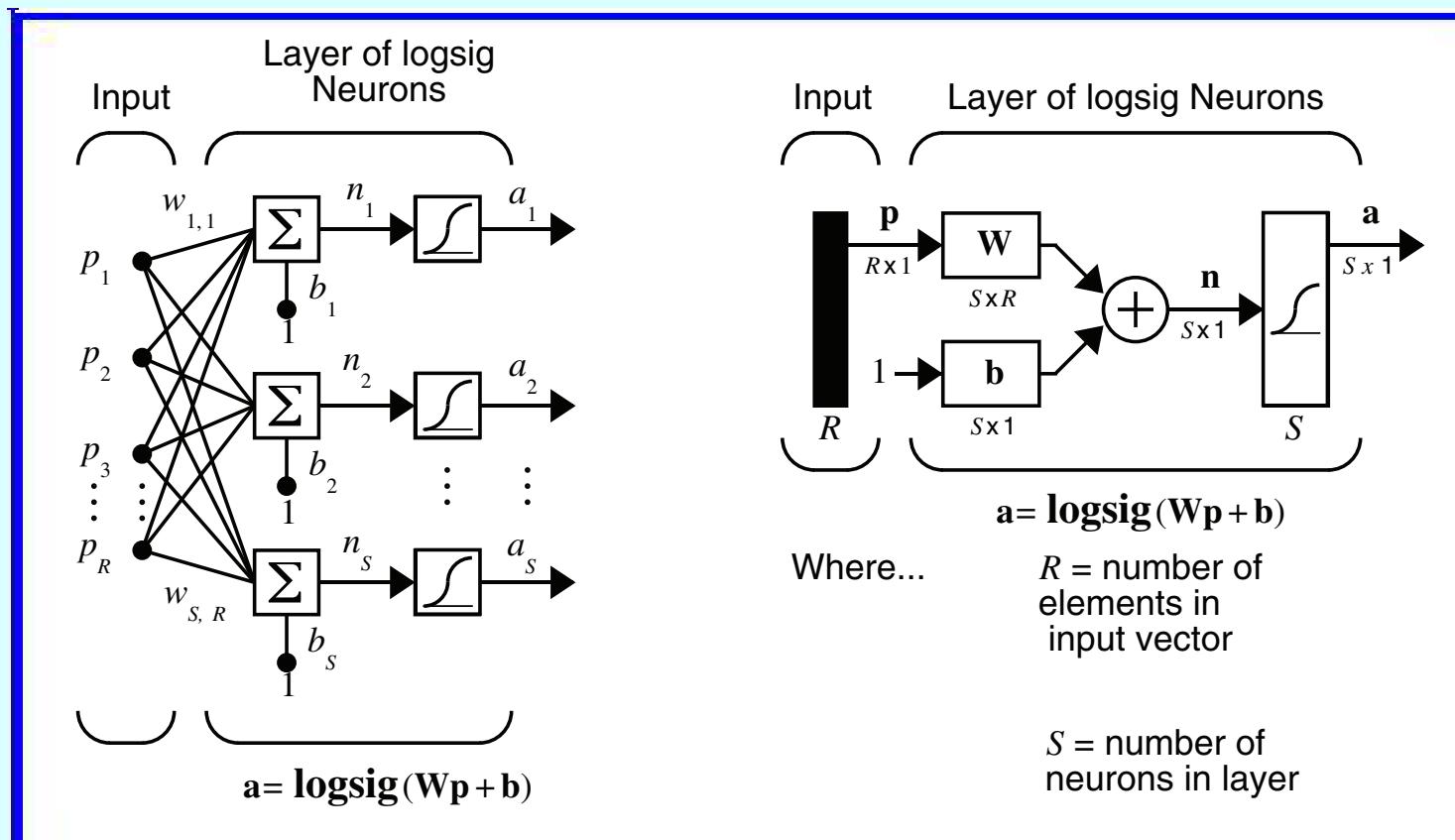
Активационные функции



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (pp. 5-8, 5-9, 8-3).

Многослойные сети прямого распространения (IV)

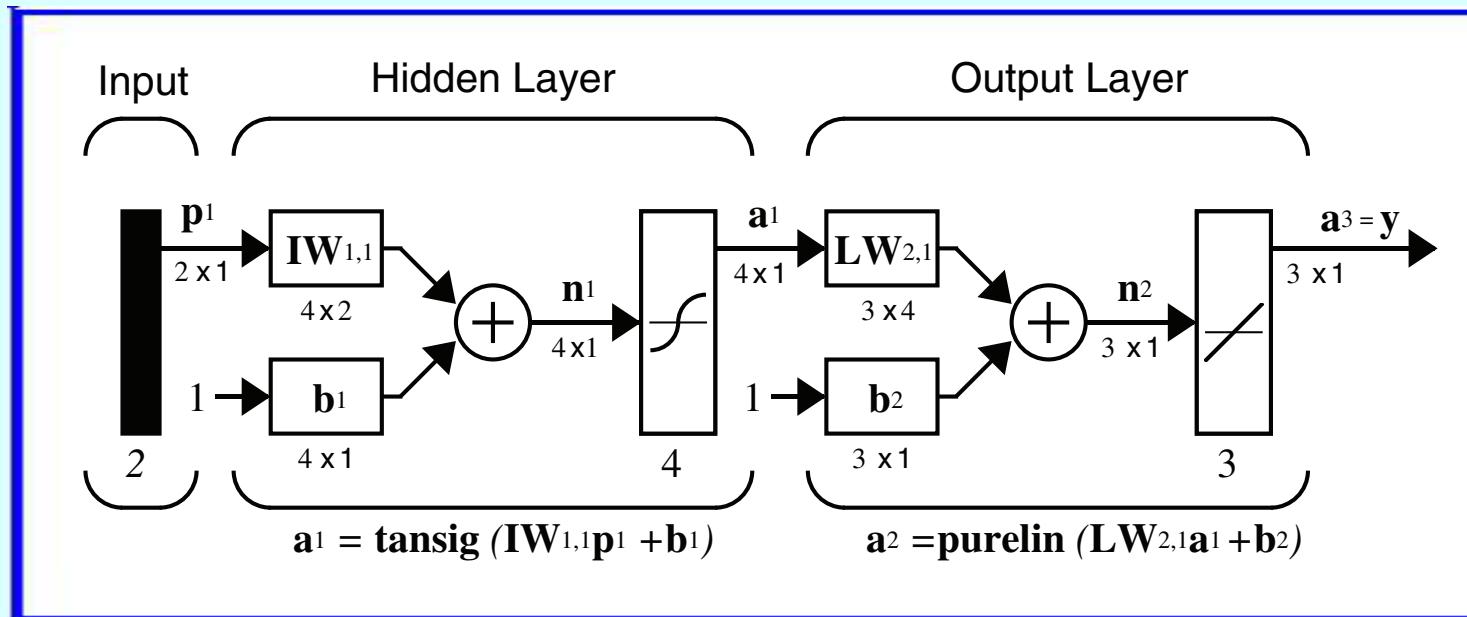
Однослойная нелинейная сеть



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-10).

Многослойные сети прямого распространения (V)

Нелинейная сеть со скрытым слоем



Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-11).

Обучение сетей прямого распространения (I)

Обучение многослойных сетей

Предыстория

Правило обучения персептрана ([Розенблatt](#)) и алгоритм LMS ([Уидроу-Хофф](#)) — только **однослоиные сети**, только **линейная разделяющая граница** в задачах классификации.

Многослойные сети + произвольная разделяющая граница

Пол Вербос (Paul Werbos) (1974): **алгоритм обратного распространения ошибки** (error backpropagation) для обучения **многослойных** сетей.

Werbos P. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. Thesis, Harvard University, Cambridge, MA, 1974.

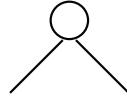
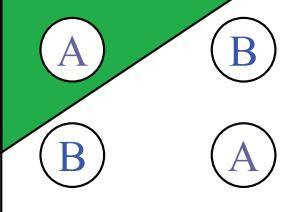
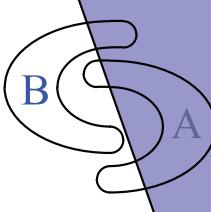
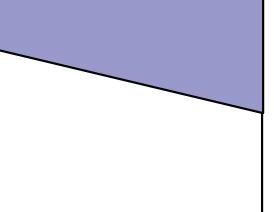
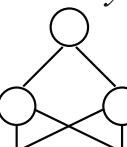
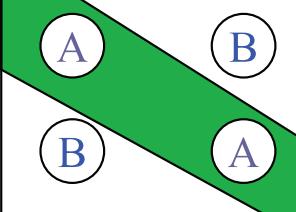
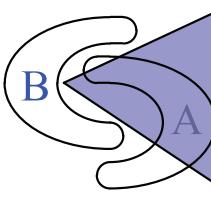
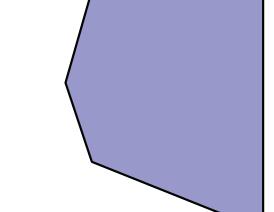
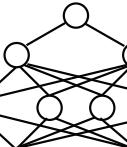
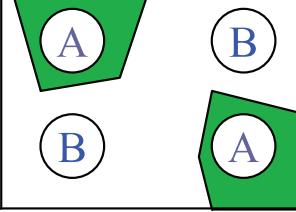
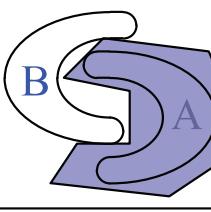
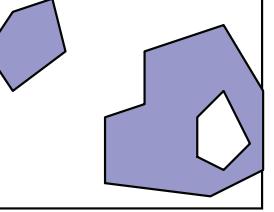
Дэвид Румельхарт (David Rumelhart), **Джеффри Хинтон** (Geoffrey) и **Рональд Уильямс** (Ronald Williams) (1986): **современный вариант** алгоритма обратного распространения ошибки для обучения многослойных сетей.

Rumelhart D.L., Hinton G.E., Williams R.J. Learning representations by back-propagating errors // *Nature*. – 1986. – Vol. 323, No. 6088. – pp. 533–536.

Rumelhart D.L., McClelland J.L. (Eds.) Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1, Cambridge, MA: MIT Press, 1986.

Обучение сетей прямого распространения (II)

Проблема разделимости классов – 1

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

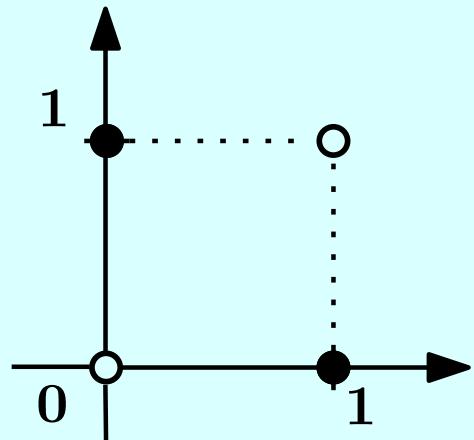
Источник: [Lippman R.P.](#) An introduction to computing with neural net // Acoustics, Speech and Signal Processing Magazine. – April 1987. – Vol 3, No. 4. – pp.4–22 (Figure 14, p. 14).

Обучение сетей прямого распространения (III)

Проблема разделимости классов – 2

Логическая функция XOR (исключающее ИЛИ):

Канонический пример задачи, недоступной
для однослойных сетей

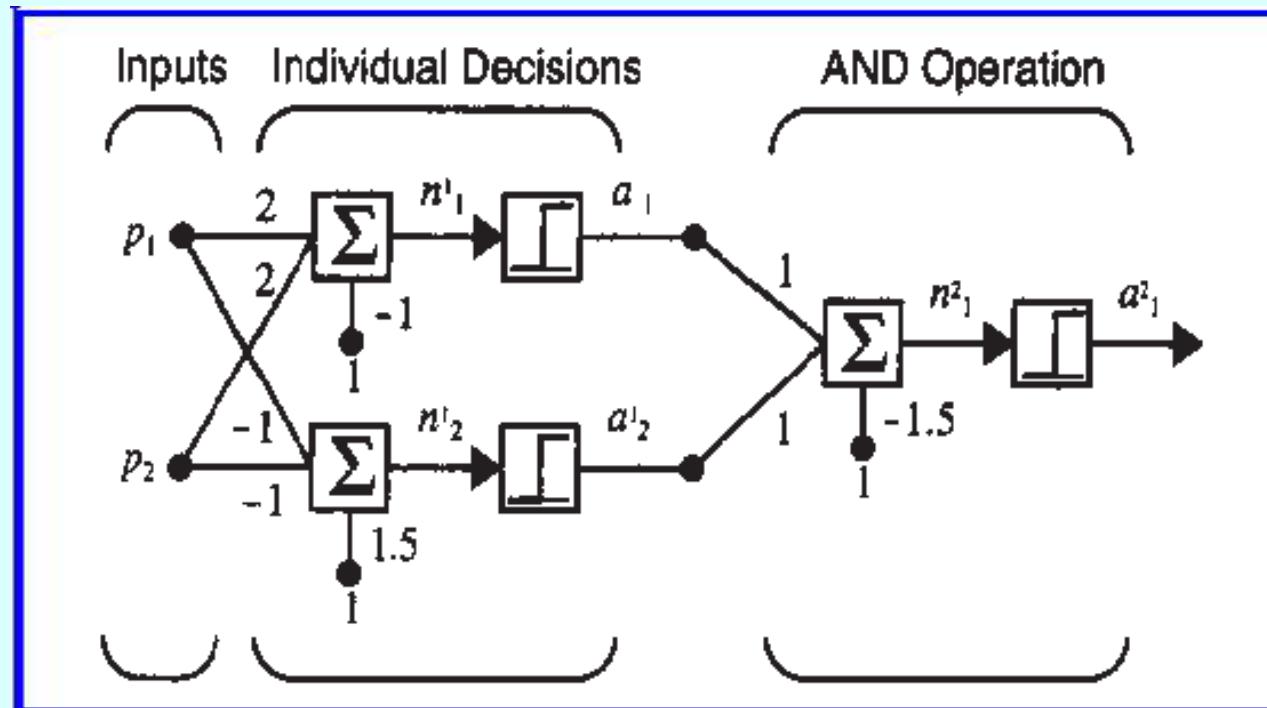


$$\begin{cases} p_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_1 = 0 \end{cases} \quad \begin{cases} p_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 1 \end{cases}$$
$$\begin{cases} p_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 1 \end{cases} \quad \begin{cases} p_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 0 \end{cases}$$

Обучение сетей прямого распространения (V)

Проблема разделимости классов – 3

Решение задачи XOR с помощью двухслойной нелинейной сети (1)

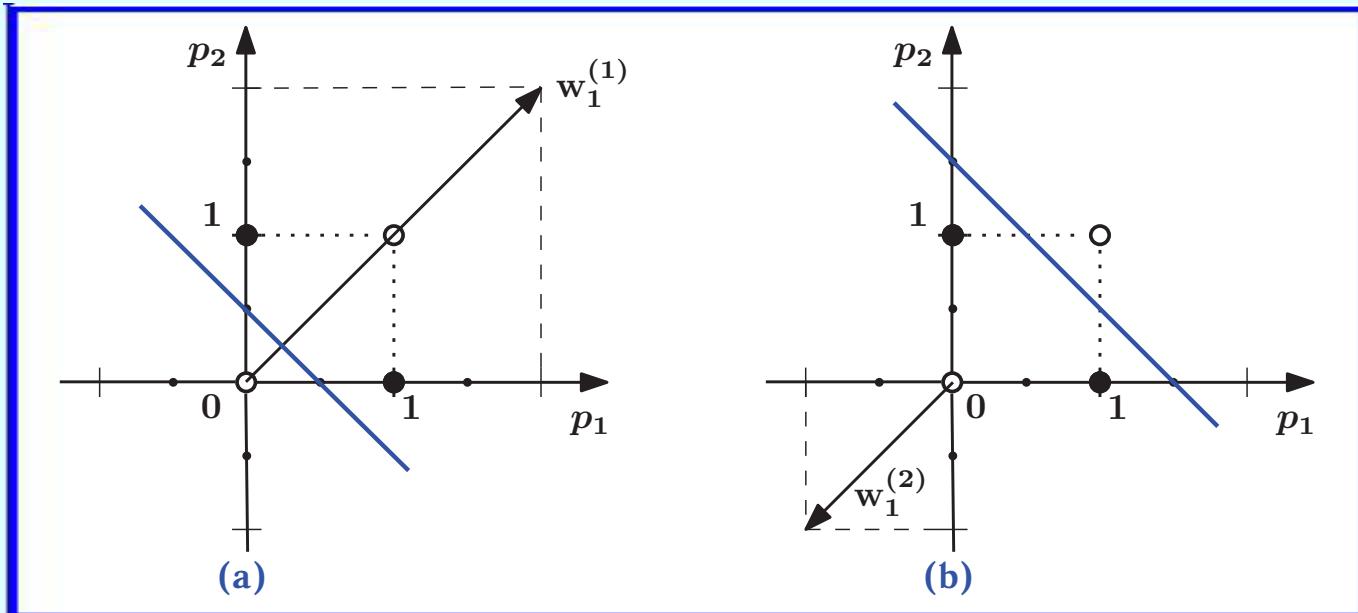


Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.3, pp.11-4).

Обучение сетей прямого распространения (VI)

Проблема разделимости классов – 4

Решение задачи XOR с помощью двухслойной нелинейной сети (2)



(a) разделяющая граница, формируемая **первым** нейроном скрытого слоя
(первый нейрон «изолирует» точку **(0,0)**)

$$n_1^{(1)} = 2p_1 + 2p_2 - 1, \quad a_1^{(1)} = \text{hardlim}(n_1^{(1)}) = \begin{cases} 1, & n_1^{(1)} \geq 0; \\ 0, & n_1^{(1)} < 0. \end{cases}$$

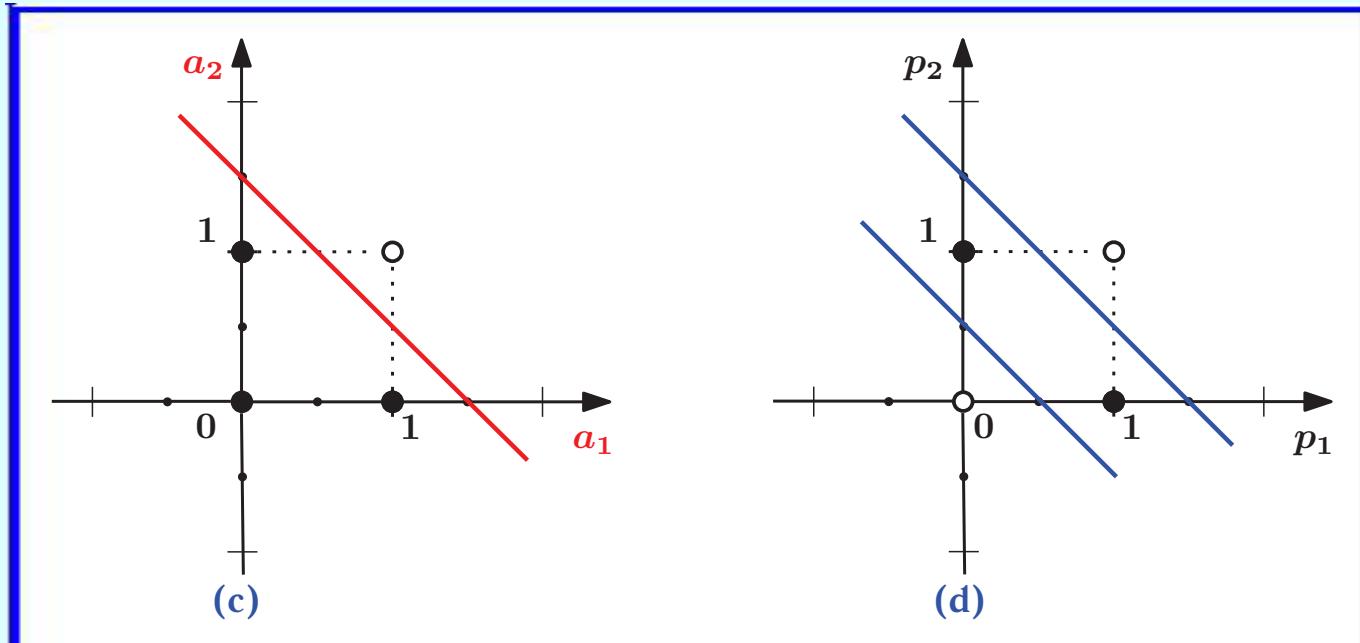
(b) разделяющая граница, формируемая **вторым** нейроном скрытого слоя
(второй нейрон «изолирует» точку **(1,1)**)

$$n_1^{(2)} = -p_1 - p_2 + 1.5, \quad a_1^{(2)} = \text{hardlim}(n_1^{(2)}) = \begin{cases} 1, & n_1^{(2)} \geq 0; \\ 0, & n_1^{(2)} < 0. \end{cases}$$

Обучение сетей прямого распространения (VII)

Проблема разделимости классов – 5

Решение задачи XOR с помощью двухслойной нелинейной сети (3)



(c) разделяющая граница, формируемая нейроном **выходного** слоя
(нейрон выходного слоя выполняет операцию **AND** для выходов нейронов скрытого слоя)

$$n_1^{(2)} = a_1^{(1)} + a_2^{(1)} - 1.5, \quad a_1^{(2)} = \text{hardlim}(n_1^{(2)}) = \begin{cases} 1, & n_1^{(2)} \geq 0; \\ 0, & n_1^{(2)} < 0. \end{cases}$$

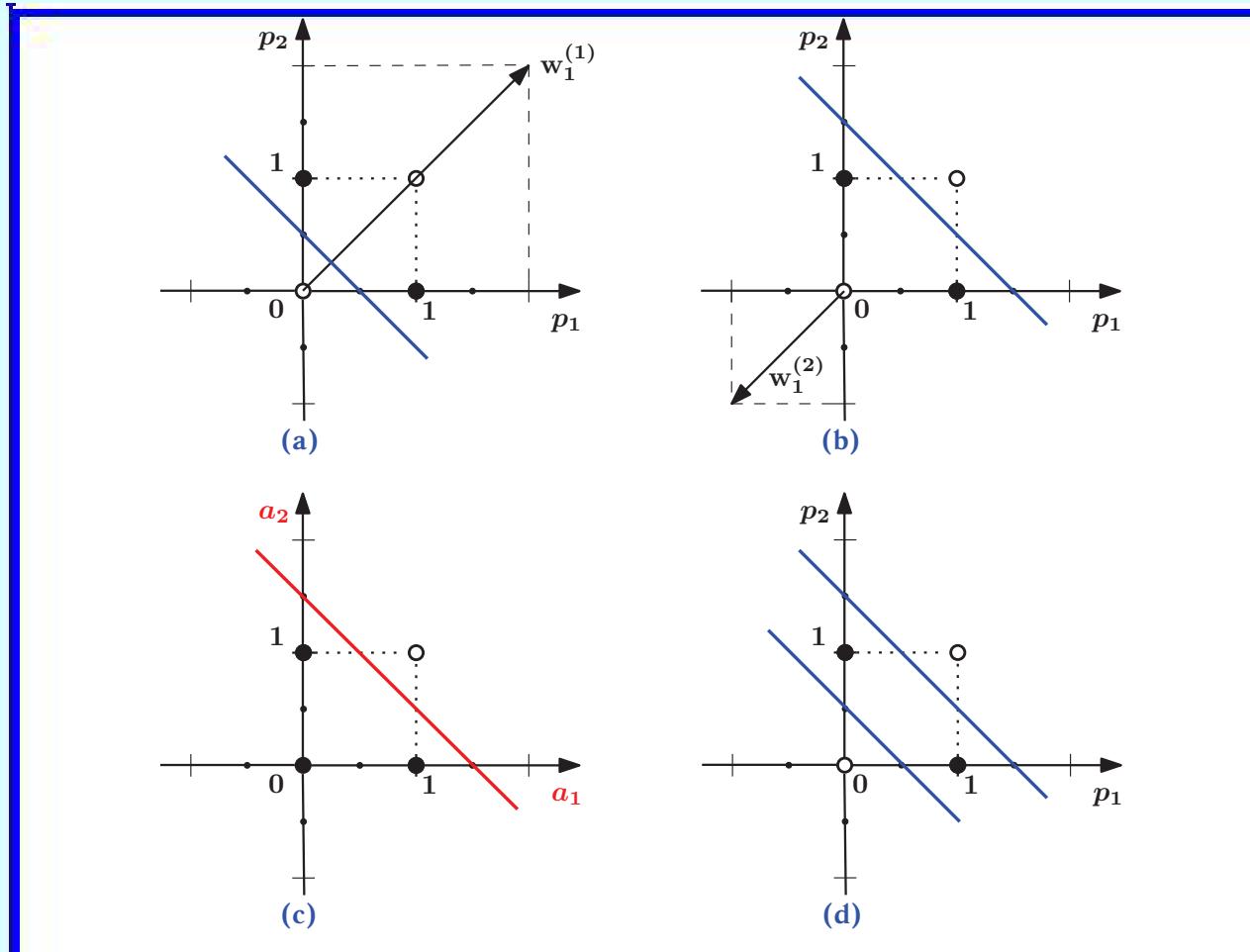
$$\text{AND} : (0, 0), (0, 1), (1, 0) \Rightarrow 0; \quad (1, 1) \Rightarrow 1$$

(d) разделение признакового пространства на области, формируемое **сетью в целом**

Обучение сетей прямого распространения (VIII)

Проблема разделимости классов – 6

Решение задачи XOR с помощью двухслойной нелинейной сети (4)



Обучение сетей прямого распространения (IX)

Проблема разделимости классов – 7

Решение задачи XOR с помощью двухслойной нелинейной сети (5)

Проверка работоспособности сети

Обучающий набор:

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_1 = 0 \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 1 \right\}$$

$$\left\{ \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 1 \right\} \quad \left\{ \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 0 \right\}$$

Матрицы весов связей между слоями сети:

$$\mathbf{W}^{(1)} = \begin{bmatrix} 2 & 2 \\ -1 & -1 \end{bmatrix}, \mathbf{b}^{(1)} = \begin{bmatrix} -1 \\ 1.5 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} 1 & 1 \end{bmatrix}, b^{(2)} = -1.5$$

Преобразования, выполняемые сетью для входного вектора $\mathbf{p} = (p_1, p_2)^T$:

$$a_1^{(1)} = \text{hardlim}(2p_1 + 2p_2 - 1)$$

$$a_2^{(1)} = \text{hardlim}(-p_1 - p_2 + 1.5)$$

$$a^{(2)} = \text{hardlim}(a_1^{(1)} - a_2^{(1)} - 1.5)$$

Обучение сетей прямого распространения (X)

Проблема разделимости классов – 8

Решение задачи XOR с помощью двухслойной нелинейной сети (6)

Проверка работоспособности сети

$$a_1^{(1)} = \text{hardlim}(2p_1 + 2p_2 - 1)$$

$$a_2^{(1)} = \text{hardlim}(-p_1 - p_2 + 1.5)$$

$$a^{(2)} = \text{hardlim}(a_1^{(1)} - a_2^{(1)} - 1.5)$$

Точка $(0, 0) \rightarrow t_1 = 0$

$$a_1^{(1)} = \text{hardlim}(2 \cdot 0 + 2 \cdot 0 - 1) = \text{hardlim}(-1) = 0$$

$$a_2^{(1)} = \text{hardlim}((-1) \cdot 0 + (-1) \cdot 0 + 1.5) = \text{hardlim}(1.5) = 1$$

$$a^{(2)} = \text{hardlim}(1 \cdot 0 + 1 \cdot 1 - 1.5) = \text{hardlim}(-0.5) = 0 = t_1$$

Точка $(0, 1) \rightarrow t_2 = 1$

$$a_1^{(1)} = \text{hardlim}(2 \cdot 0 + 2 \cdot 1 - 1) = \text{hardlim}(1) = 1$$

$$a_2^{(1)} = \text{hardlim}((-1) \cdot 0 + (-1) \cdot 1 + 1.5) = \text{hardlim}(0.5) = 1$$

$$a^{(2)} = \text{hardlim}(1 \cdot 1 + 1 \cdot 1 - 1.5) = \text{hardlim}(0.5) = 1 = t_2$$

Обучение сетей прямого распространения (XI)

Проблема разделимости классов – 9

Решение задачи XOR с помощью двухслойной нелинейной сети (7)

Проверка работоспособности сети

$$a_1^{(1)} = \text{hardlim}(2p_1 + 2p_2 - 1)$$

$$a_2^{(1)} = \text{hardlim}(-p_1 - p_2 + 1.5)$$

$$a^{(2)} = \text{hardlim}(a_1^{(1)} - a_2^{(1)} - 1.5)$$

Точка $(1, 0) \rightarrow t_3 = 1$

$$a_1^{(1)} = \text{hardlim}(2 \cdot 1 + 2 \cdot 0 - 1) = \text{hardlim}(1) = 1$$

$$a_2^{(1)} = \text{hardlim}((-1) \cdot 1 + (-1) \cdot 0 + 1.5) = \text{hardlim}(0.5) = 1$$

$$a^{(2)} = \text{hardlim}(1 \cdot 1 + 1 \cdot 1 - 1.5) = \text{hardlim}(0.5) = 1 = t_3$$

Точка $(1, 1) \rightarrow t_4 = 0$

$$a_1^{(1)} = \text{hardlim}(2 \cdot 1 + 2 \cdot 1 - 1) = \text{hardlim}(3) = 1$$

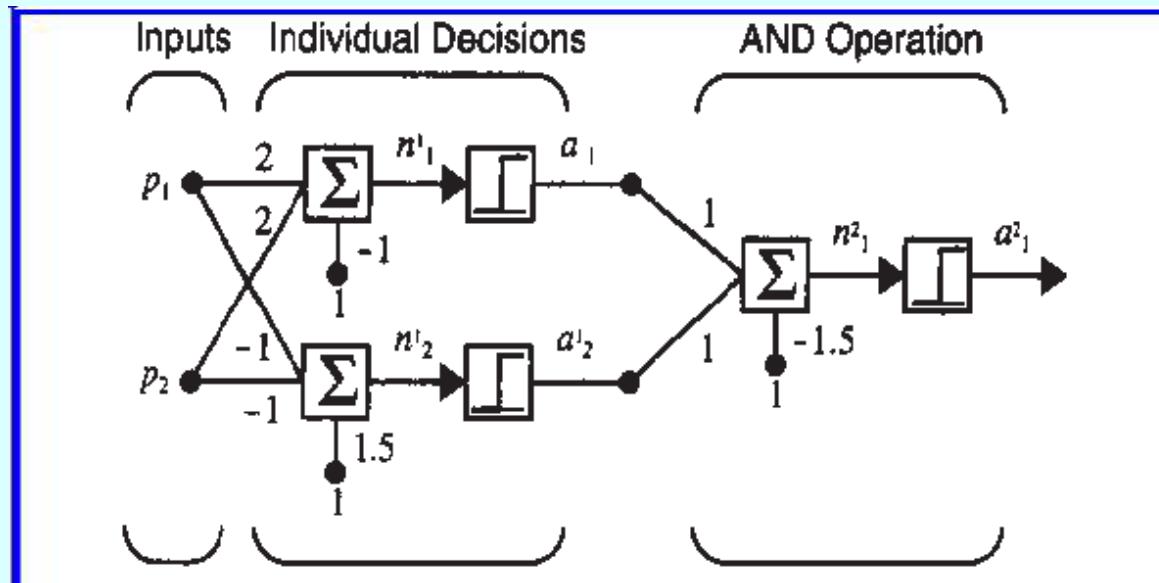
$$a_2^{(1)} = \text{hardlim}((-1) \cdot 1 + (-1) \cdot 1 + 1.5) = \text{hardlim}(-0.5) = 0$$

$$a^{(2)} = \text{hardlim}(1 \cdot 1 + 1 \cdot 0 - 1.5) = \text{hardlim}(-0.5) = 0 = t_4$$

Обучение сетей прямого распространения (XII)

Проблема разделимости классов – 10

Решение задачи XOR с помощью двухслойной нелинейной сети (8)



Скрытый слой — персепtron Розенблата
Выходной слой — ADALINE (в варианте Уидроу)

Отображения, реализуемые сетью:

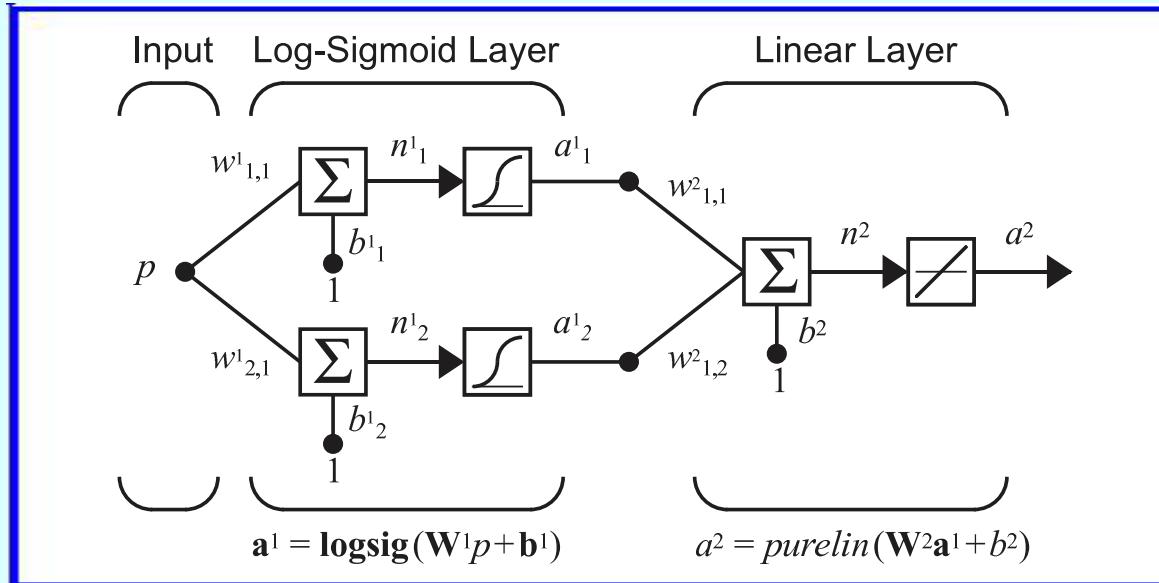
Скрытый слой — отображение $\Psi_1 : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ входного (признакового) пространства в промежуточное пространство выходов скрытого слоя, $a^{(1)} = \Psi_1(p)$.

Выходной слой — отображение $\Psi_2 : \mathbb{R}^2 \rightarrow \mathbb{R}^1$ промежуточного пространства выходов скрытого слоя в пространство выходов сети, $a^{(2)} = \Psi_2(a^{(1)})$.

Сеть в целом — композиция $(\Psi_1 \circ \Psi_2)$ отображений Ψ_1 и Ψ_2 , т. е.
 $a^{(2)} = \Psi_2(\Psi_1(p))$.

Обучение сетей прямого распространения (XII)

Задача аппроксимации функций



$$a_1^{(1)} = \text{logsig}(w_{1,1}^{(1)}p + b_1^{(1)})$$

$$a_2^{(1)} = \text{logsig}(w_{2,1}^{(1)}p + b_2^{(1)})$$

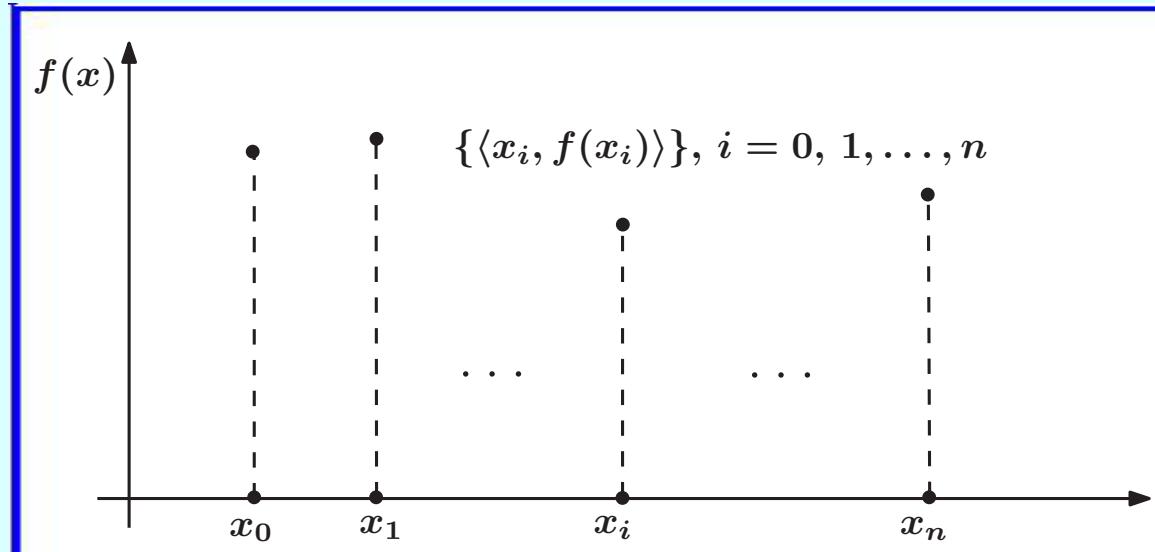
$$a^{(2)} = \text{purelin}(w_{1,1}^{(2)}a_1^{(1)} - w_{1,2}^{(2)}a_2^{(1)} + b_2)$$

$$\text{logsig}(n) = \frac{1}{1 + e^{-n}}, \quad \text{purelin}(n) = n$$

Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.4, pp.11-5).

Обучение сетей прямого распространения (XIII)

Приближенное представление зависимостей – 1



Имеется **таблично заданная** функция

$$f(x) = \{\langle x_i, f(x_i) \rangle\}, i = 1, \dots, n$$

Ее надо **представить приближенно** с помощью некоторой другой функции $\hat{f}(x)$.

Обучение сетей прямого распространения (XIV)

Приближенное представление зависимостей – 2

Каким образом следует выбирать функцию $\hat{f}(x)$ среди возможных вариантов?

Откуда взять сам набор $\{\hat{f}(x)\}$ этих возможных вариантов?

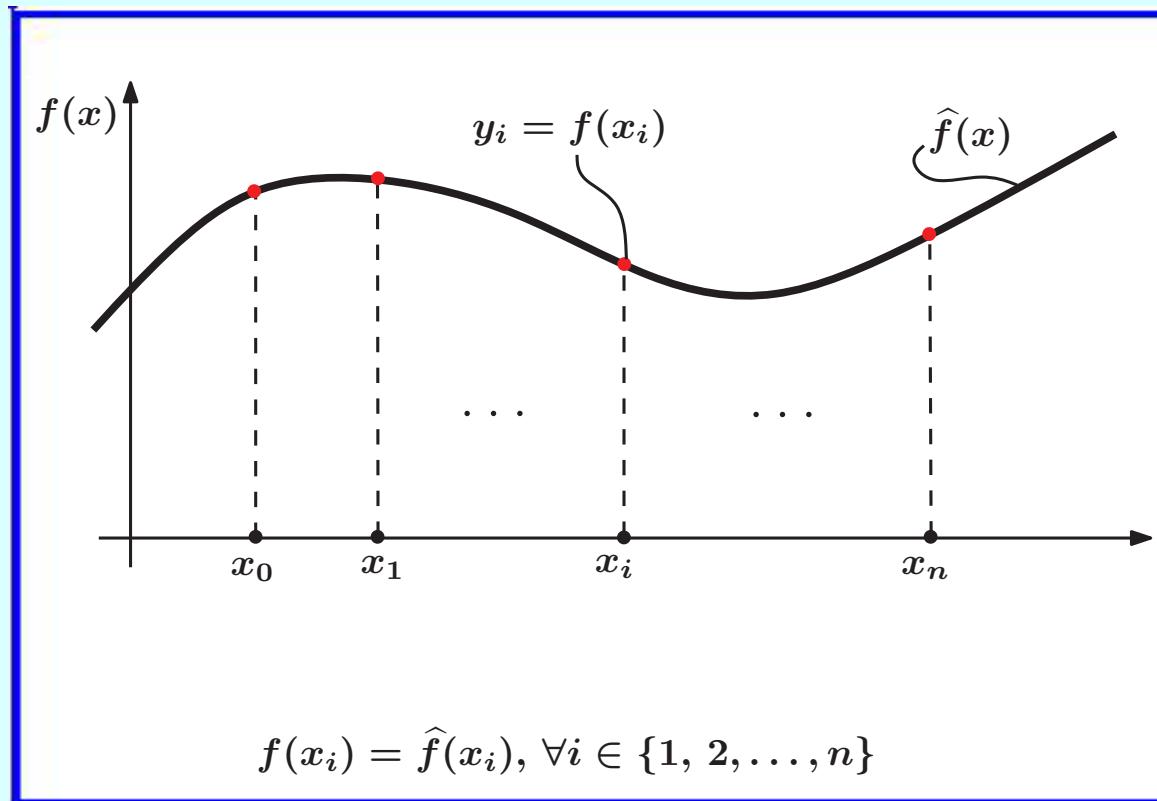
Для первого вопроса (**как выбирать функцию $\hat{f}(x)$**), есть **два основных варианта** ответа:

- из условия **совпадения** $f(x)$ и $\hat{f}(x)$ в узлах сетки $\Delta_n : x_0 < x_1 < \dots < x_n$ (это задача *интерполяции* или *экстраполяции*);
- из условия **минимума расстояния** между функциями $f(x)$ и $\hat{f}(x)$ (это задача *аппроксимации* или *сглаживания*).

Наиболее важным для практики является **второй** из этих двух вариантов.

Обучение сетей прямого распространения (XV)

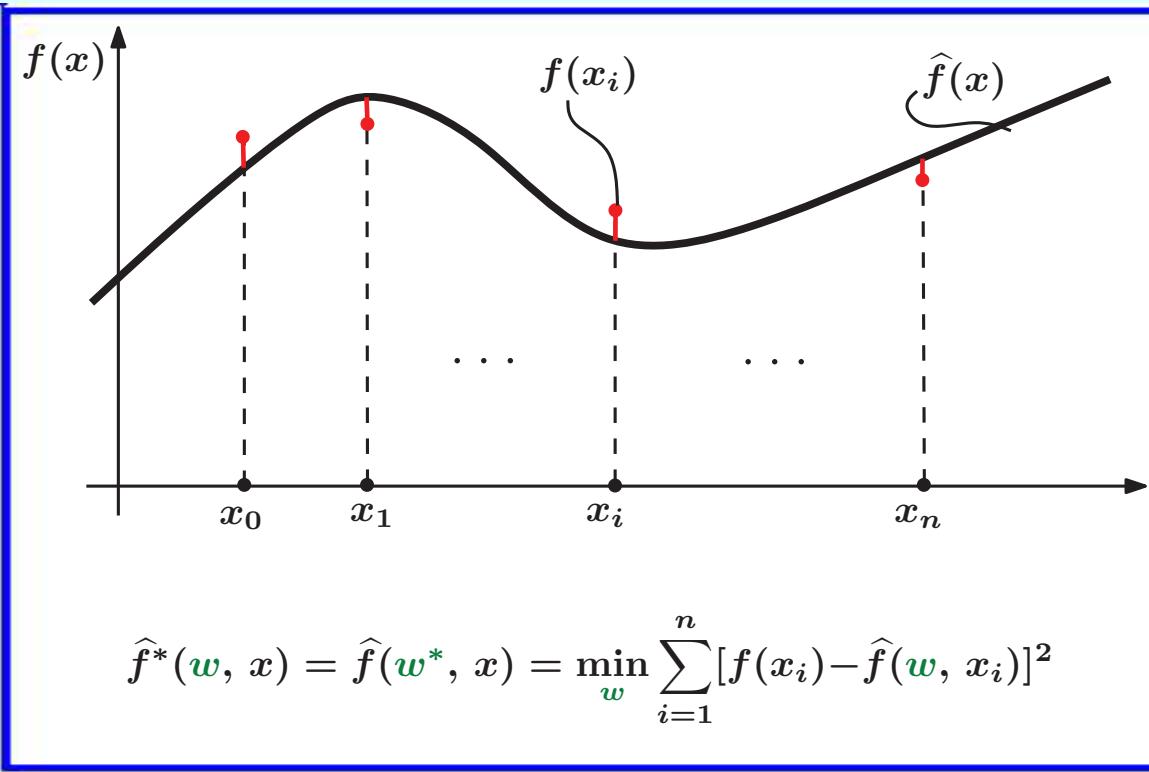
Приближенное представление зависимостей – 3



Задача интерполяции решается из условия **совпадения** $f(x)$ и $\hat{f}(x)$
в узлах сетки Δ_n : $x_0 < x_1 < \dots < x_n$.

Обучение сетей прямого распространения (XVI)

Приближенное представление зависимостей – 4

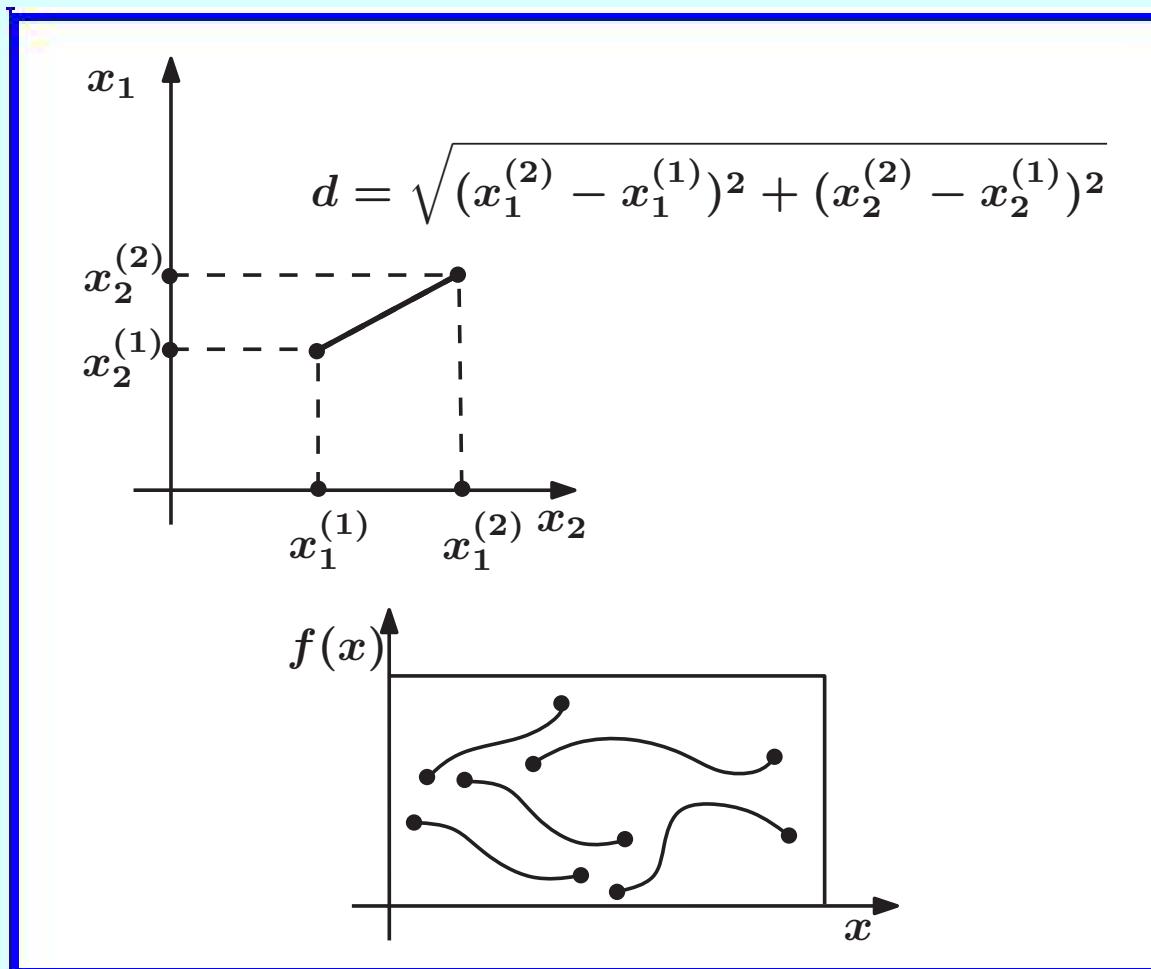


Задача аппроксимации решается из условия **минимума расстояния** между функциями $f(x)$ и $\hat{f}(x)$.

Чтобы найти функцию $\hat{f}(x)$, **наиболее близкую** к функции $f(x)$, следует уточнить **понятие расстояния** между этими функциями.

Обучение сетей прямого распространения (XVII)

Расстояние между функциями – 1



Обучение сетей прямого распространения (XVIII)

Расстояние между функциями – 2

В прикладных задачах чаще всего используются следующие **два основных варианта** определения расстояния:

1. Первый из них основан на **равномерной норме**

$$||h(t)||_C = \max_{x \in X} |h(t)|$$

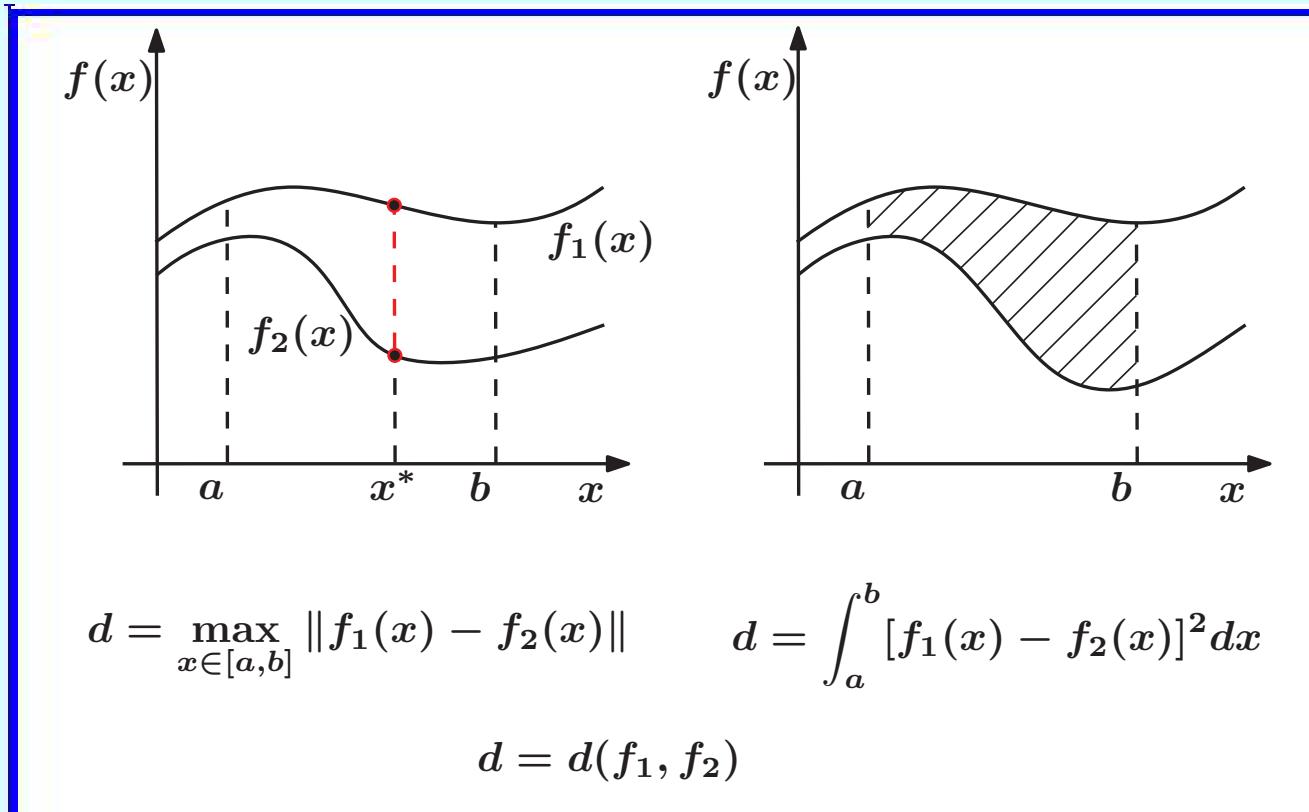
2. Второй основан на **среднеквадратичной норме**

$$||h(t)||_{L_2} = \left(\int_X |h(t)|^2 dX \right)^{1/2}$$

Здесь $h(x) = f(x) - g(x)$.

Обучение сетей прямого распространения (XIX)

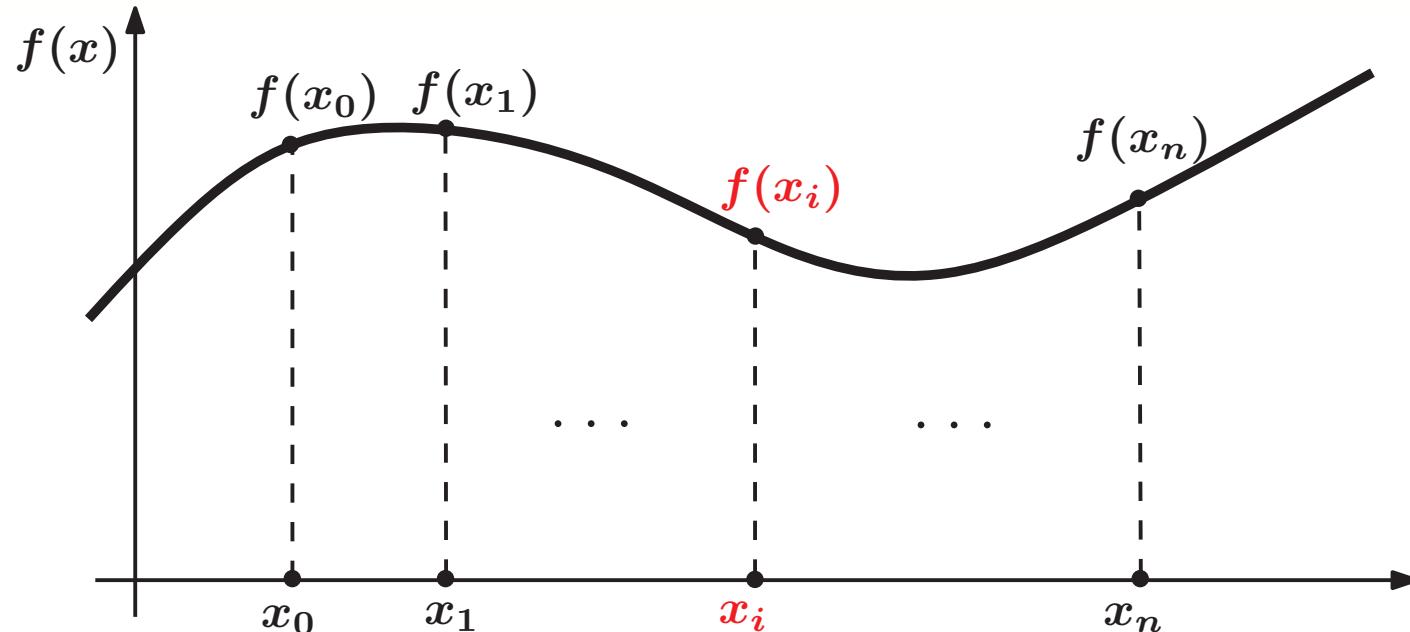
Расстояние между функциями – 3



$$g(x) = |f_1(x) - f_2(x)| \Rightarrow \max_{x \in [a,b]} g(x); \quad g(x) = |f_1(x) - f_2(x)| \Rightarrow \int_a^b g(x) dx$$

Обучение сетей прямого распространения (ХХ)

Формирование обучающего набора

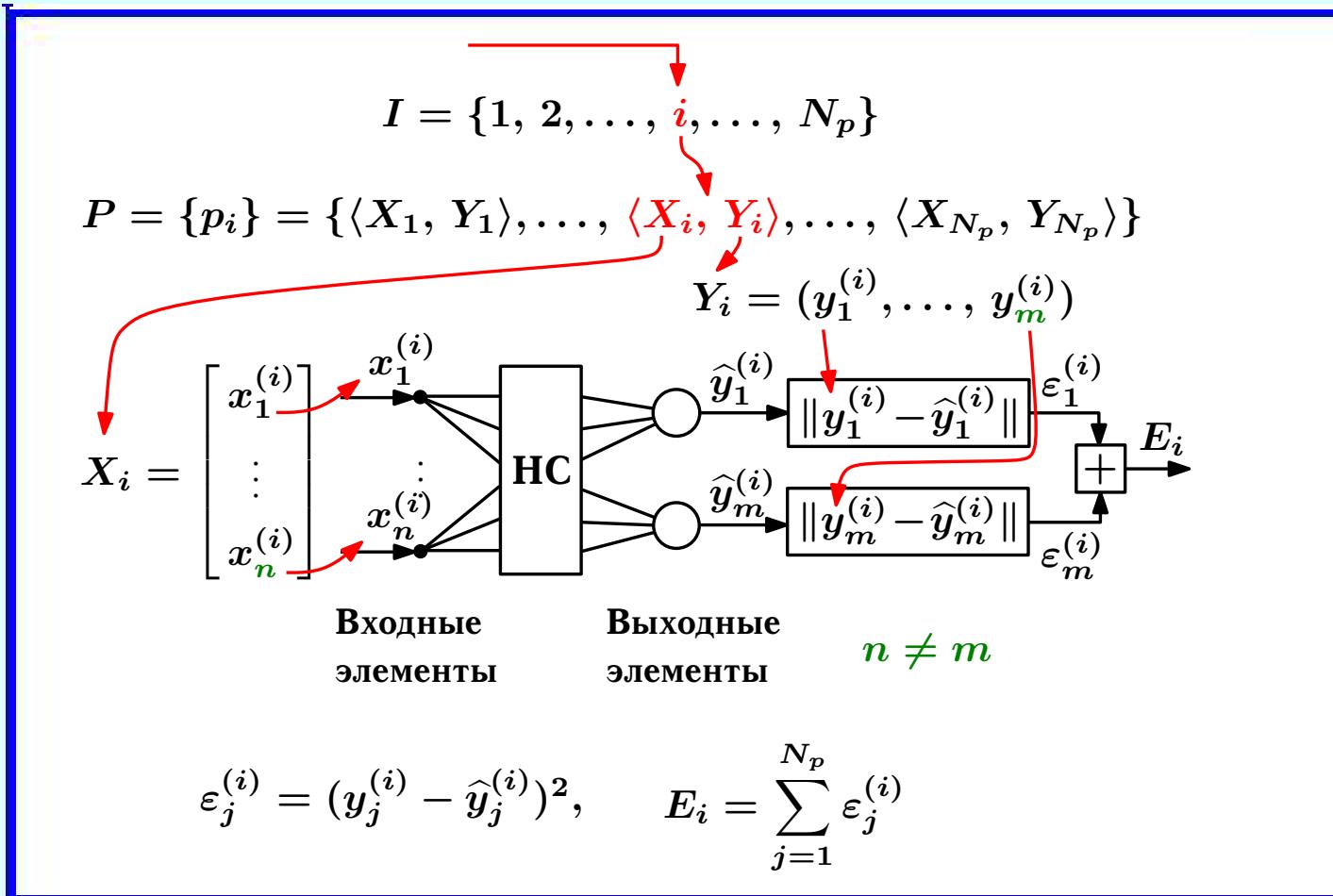


$$P = \{\langle x_0, f(x_0) \rangle, \langle x_1, f(x_1) \rangle, \dots, \langle x_i, f(x_i) \rangle, \dots, \langle x_n, f(x_n) \rangle\}$$

$$P = \{p_i\} = \{\langle x_i, f(x_i) \rangle\}, i = 0, 1, \dots, n$$

Обучение сетей прямого распространения (ХХI)

Взаимодействие обучающего набора и НС-модели



Здесь i — номер **текущего примера** из обучающего набора (задачника),
 E_i — **ошибка** НС-модели на этом примере.

Обучение сетей прямого распространения (XXII)

**Нейросеть прямого распространения
как аппроксиматор отображений – 1**

Нейросеть прямого распространения, содержащая скрытые слои — мощное аппроксимирующее средство.

В частности, с помощью **трехслойной НС**, в которой скрытые слои содержат элементы с сигмоидальными функциями, а выходной слой — элементы с линейной активационной функцией, можно **с любой наперед заданной точностью** аппроксимировать **любое непрерывное отображение** n -мерного вектора \mathbf{X} в m -мерный вектор \mathbf{Y} .

Обучение сетей прямого распространения (ХХIII)

Нейросеть прямого распространения как аппроксиматор отображений – 2

Для решения задачи аппроксимации отображений необходимо соответственно выбрать **структурные признаки сети**:

- число слоев** в сети;
- число элементов** в каждом слое;
- вид** этих элементов.

Совокупность структурных признаков называют **архитектурой нейросети**.

Параметрами, которые можно варьировать в НС — это обычно ее **синаптические веса**. Их следует подобрать таким образом, чтобы **минимизировать ошибку аппроксимации** для требуемого отображения.

Аппроксирующие свойства НС зависят от вида и количества элементов-нейронов, а также от структуры связей между ними.

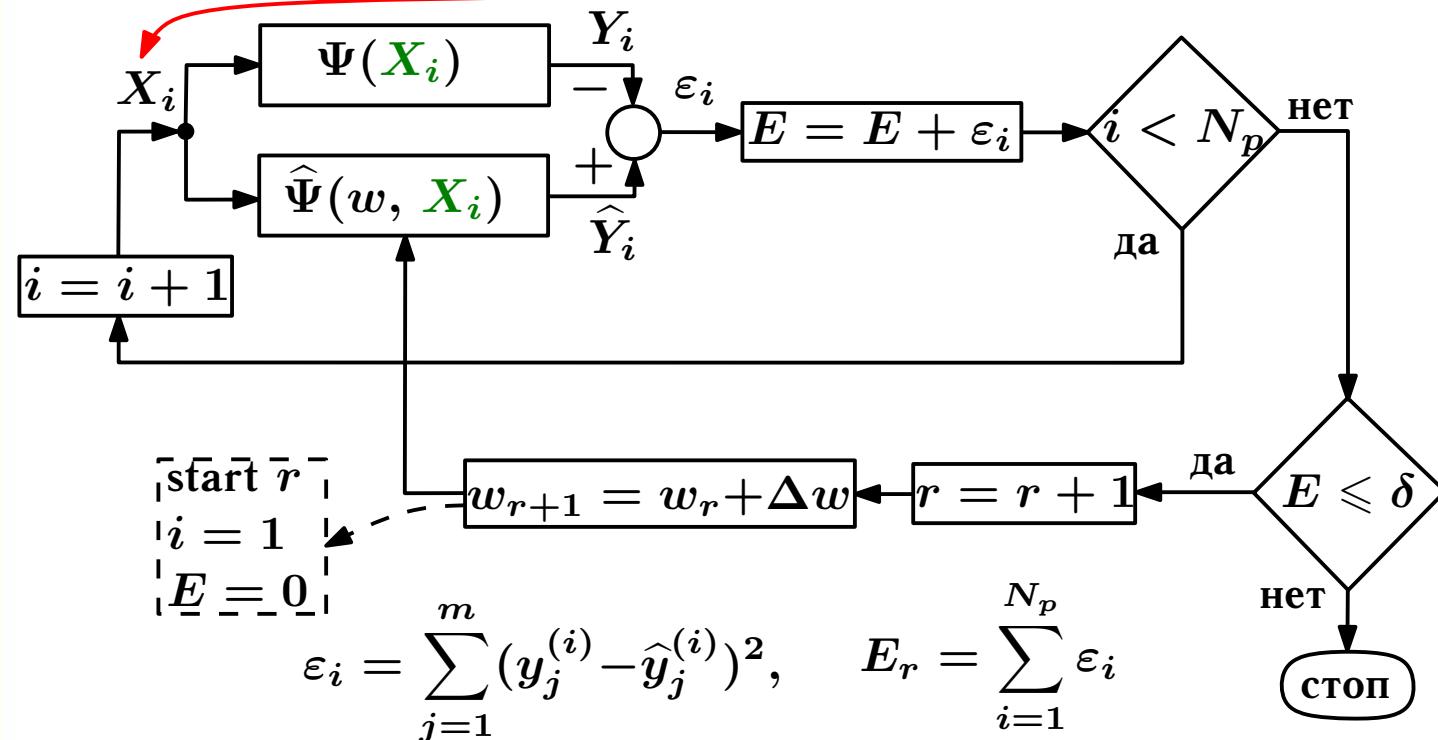
Обучение сетей прямого распространения (XXIV)

Апроксимация НС-моделью отображения $\Psi : \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$Y_i = \Psi(X_i), i = 1, \dots, N_p, X_i \in \mathbb{R}^n, Y_i \in \mathbb{R}^m, \Psi : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

$$X_i = (x_1^{(i)}, \dots, x_n^{(i)}), \quad Y_i = (y_1^{(i)}, \dots, y_m^{(i)}) \quad n \neq m$$

$$P = \{p_i\} = \{\langle X_1, Y_1 \rangle, \dots, \langle X_{N_p}, Y_{N_p} \rangle\}$$



Обучение сетей прямого распространения (XXV)

Общая схема процесса обучения нейросети – 1

Подбор значений синаптических весов будем выполнять с помощью процесса **обучением с учителем**.

Суть процесса обучения. Пусть имеется НС, зададим каким-либо образом **начальные значения** синаптических весов **W** в ней (чаще всего — как небольшие случайные числа, равномерно распределенные на отрезке **[0, 1]**). Подадим на **вход** НС некоторый набор сигналов

$$\mathbf{X}^{(\varepsilon)} = (X_1^{(\varepsilon)}, \dots, X_N^{(\varepsilon)}).$$

Распространяясь по сети, этот набор порождает набор **выходных сигналов**

$$\mathbf{Y} = (Y_1, \dots, Y_M).$$

Значения выходного вектора **Y** сети зависят от значений **входного вектора X^(ε)** и от значений **синаптических весов W**.

Обучение сетей прямого распространения (XXVI)

Общая схема процесса обучения нейросети – 2

Пусть для данного значения $\mathbf{X}^{(e)}$ **входного вектора \mathbf{X}** известно, каким должно быть «**правильное**» («желаемое») значение $\mathbf{Y}^{(e)}$ **выходного вектора \mathbf{Y}** (именно это значение и предъявляется «учителем» на соответствующем шаге процесса обучения).

Найдем **уклонение** желаемого (требуемого) выхода $\mathbf{Y}^{(e)}$ от того выхода \mathbf{Y} сети, который получился для данного $\mathbf{X}^{(e)}$ при некотором фиксированном значении синаптических весов \mathbf{W} ; оно для данного входного вектора $\mathbf{X}^{(e)}$ представляет собой **ошибку**, являющуюся функцией от параметров \mathbf{W} :

$$\mathcal{E}(\mathbf{W}) = \|\mathbf{Y}^{(e)} - \mathbf{Y}(\mathbf{X}^{(e)}, \mathbf{W})\|$$

Обучение сетей прямого распространения (XXVII)

Общая схема процесса обучения нейросети – 3

Упорядоченную пару векторов $\mathbf{p} = \langle \mathbf{X}^{(\varepsilon)}, \mathbf{Y}^{(\varepsilon)} \rangle$, которая использовалась для получения выражения

$$\mathcal{E}(W) = ||\mathbf{Y}^{(\varepsilon)} - \mathbf{Y}(\mathbf{X}^{(\varepsilon)}, W)||$$

принято называть **обучающим примером**, а набор из N_P таких пар

$$P = \{p_k\} = \{\langle \mathbf{X}_k^{(\varepsilon)}, \mathbf{Y}_k^{(\varepsilon)} \rangle\}, k = 1, \dots, N_P$$

именуется **обучающим набором**.

Обучение сетей прямого распространения (XXVIII)

Общая схема процесса обучения нейросети – 4

Функция ошибки $\mathcal{E}(\cdot)$ может быть определена как **на отдельном обучающем примере** $p \in P$

$$\mathcal{E} = \mathcal{E}(p, W),$$

так и **на обучающем наборе** $P = \{p_i\}$ в целом

$$\mathcal{E} = \mathcal{E}(P, W).$$

Обучение сетей прямого распространения (XXIX)

Общая схема процесса обучения нейросети – 5

Задача обучения сети — это задача **минимизации функции ошибки** по синаптическим весам **W**:

$$\mathcal{E}(W^*) = \min_W \mathcal{E}(p, W),$$

для случая использования **отдельных обучающих примеров** или же

$$\mathcal{E}(W^*) = \min_W \mathcal{E}(P, W),$$

для случая использования **обучающего набора в целом**.

Обучение сетей прямого распространения (XXX)

Общая схема процесса обучения нейросети – 6

Резюме

Зададим **начальные значения** синаптических весов \mathbf{W} в сети. Подадим на **вход** сети некоторый набор сигналов $\mathbf{X}^{(\varepsilon)} = (X_1^{(\varepsilon)}, \dots, X_N^{(\varepsilon)})$. Этот набор порождает набор **выходных сигналов** $\mathbf{Y} = (Y_1, \dots, Y_M)$.

Найдем **уклонение** желаемого (требуемого) выхода $\mathbf{Y}^{(\varepsilon)}$ от полученного \mathbf{Y} ; оно для данного входного вектора $\mathbf{X}^{(\varepsilon)}$ представляет собой **ошибку**, являющуюся функцией от набора параметров \mathbf{W} :

$$\mathcal{E}(\mathbf{W}) = \|\mathbf{Y}^{(\varepsilon)} - \mathbf{Y}(\mathbf{X}^{(\varepsilon)}, \mathbf{W})\|.$$

Упорядоченная пара векторов $\mathbf{p} = \langle \mathbf{X}^{(\varepsilon)}, \mathbf{Y}^{(\varepsilon)} \rangle$ – **обучающий пример**, а набор из N_P таких пар – **обучающий набор**:

$$\mathbf{P} = \{\mathbf{p}_k\} = \{\langle \mathbf{X}_k^{(\varepsilon)}, \mathbf{Y}_k^{(\varepsilon)} \rangle\}, k = 1, \dots, N_P.$$

Функция ошибки $\mathcal{E}(\cdot)$ на обучающем наборе:

$$\mathcal{E} = \mathcal{E}(\mathbf{P}, \mathbf{W}).$$

Задача обучения сети:

$$\mathcal{E}(\mathbf{W}^*) = \min_{\mathbf{W}} \mathcal{E}(\mathbf{P}, \mathbf{W})$$

Обучение сетей прямого распространения (XXXI)

Метод обратного распространения ошибки – 1

Метод обратного распространения ошибки и его многочисленные разновидности — один из наиболее часто применяемых подходов к решению задачи **обучения многослойных сетей**.

Алгоритм обратного распространения ошибки:

1. Весам сети \mathbf{W} присваиваются небольшие **начальные значения**.
2. Выбирается очередной **обучающий пример** $p_k = \langle \mathbf{X}_k^{(\varepsilon)}, \mathbf{Y}_k^{(\varepsilon)} \rangle$ из обучающего набора

$$\mathbf{P} = \{p_k\} = \{\langle \mathbf{X}_k^{(\varepsilon)}, \mathbf{Y}_k^{(\varepsilon)} \rangle, k = 1, \dots, N_P\},$$

вектор $\mathbf{X}_k^{(\varepsilon)}$ подается на вход сети.

3. Вычисляется **выход сети** $\mathbf{Y}(\mathbf{X}_k^{(\varepsilon)}, \mathbf{W})$, отвечающий входному вектору $\mathbf{X}_k^{(\varepsilon)}$ и принятым значениям весов \mathbf{W} .

Обучение сетей прямого распространения (XXXII)

Метод обратного распространения ошибки – 2

Алгоритм обратного распространения ошибки
(продолжение):

4. На основе желаемого выхода $\mathbf{Y}_k^{(\varepsilon)}$ для примера \mathbf{p}_k и реально полученного (вычисленного на предыдущем шаге) выхода $\mathbf{Y}(\mathbf{X}_k^{(\varepsilon)}, \mathbf{W})$ вычисляется **ошибка сети** $\mathcal{E}_k(\cdot)$ для данного примера:

$$\mathcal{E}_k(\mathbf{X}_k^{(\varepsilon)}, \mathbf{W}) = \|\mathbf{Y}_k^{(\varepsilon)} - \mathbf{Y}(\mathbf{X}_k^{(\varepsilon)}, \mathbf{W})\|.$$

5. Шаги 2, 3 и 4 повторяются для всех обучающих примеров \mathbf{p}_k , $k = 1, \dots, N_P$, при одном и том же значении \mathbf{W} ; эти N_P шагов составляют **одну эпоху** в процессе обучения рассматриваемой сети.

Обучение сетей прямого распространения (XXXIII)

Метод обратного распространения ошибки – 3

Алгоритм обратного распространения ошибки
(окончание):

6. Вычисляется **суммарная ошибка** сети $\mathcal{E}^{(r)}$ для данной r -й эпохи:

$$\mathcal{E}^{(r)} = \sum_{k=1}^{N_P} \mathcal{E}_k .$$

7. Веса сети \mathbf{W} корректируются так, чтобы **уменьшить ошибку** $\mathcal{E}^{(r)}(\cdot)$.
8. Шаги со 2 по 7 повторяются до тех пор, пока не будет выполнено **условие** $\mathcal{E}_k(\mathbf{W}) \leq \epsilon$, где $\epsilon > 0$ — наперед заданное малое число; процесс обучения может быть остановлен также по достижении **предельно допустимого числа эпох**.

Обучение сетей прямого распространения (XXXIV)

Метод обратного распространения ошибки – 4

Последовательность шагов алгоритма обратного распространения ошибки реализует **итерационную процедуру постепенного подбора весов \mathbf{W}** ; для некоторой произвольной итерации (эпохи) r эту процедуру можно представить выражением вида:

$$\mathbf{W}^{r+1} = \mathbf{W}^r - \eta^r \frac{\partial \mathcal{E}}{\partial \mathbf{W}},$$

или, в скалярной форме,

$$w_{ij}^{r+1} = w_{ij}^r - \eta^r \frac{\partial \mathcal{E}}{\partial w_{ij}}.$$

Здесь скаляр η^r , $0 < \eta^r < 1$, определяет **темп обучения** на шаге r .

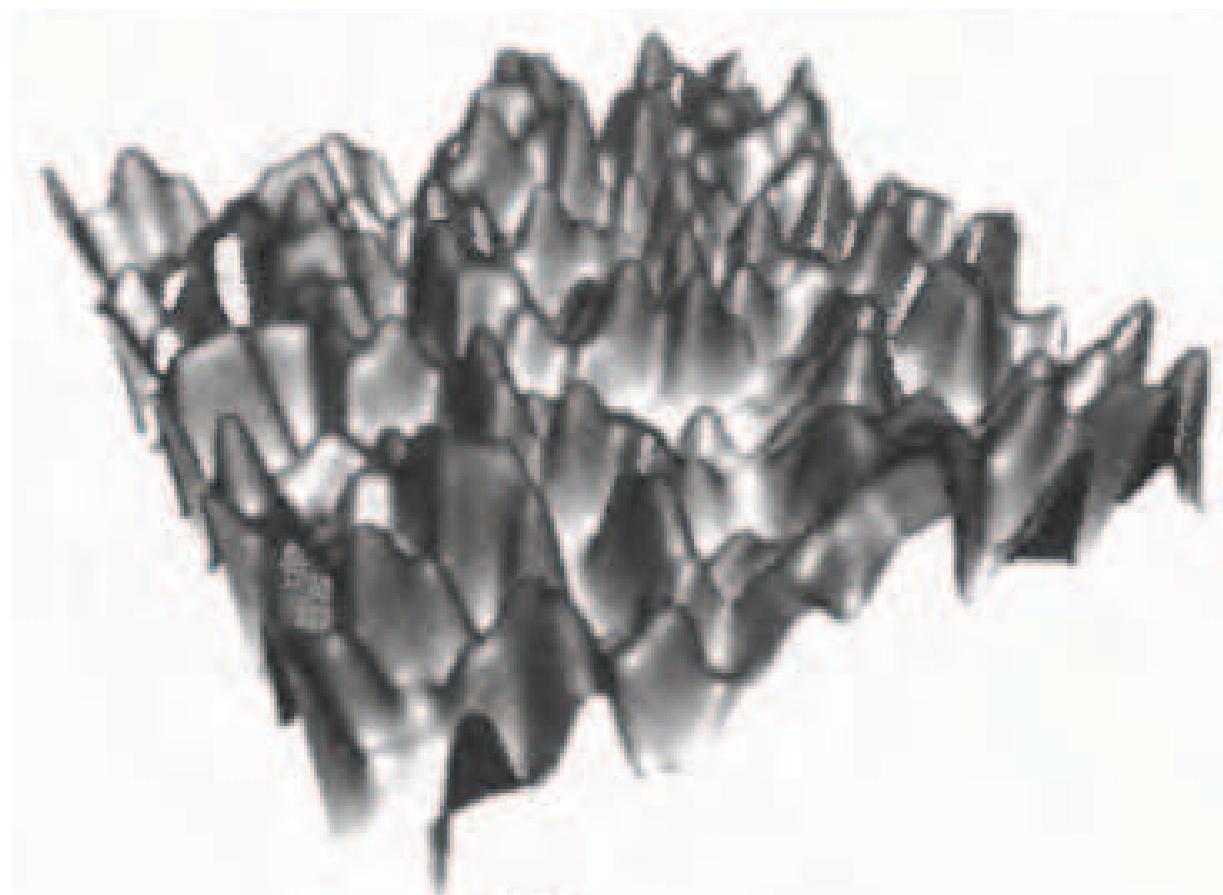
Обучение сетей прямого распространения (XXXV)

Рельеф функции ошибки



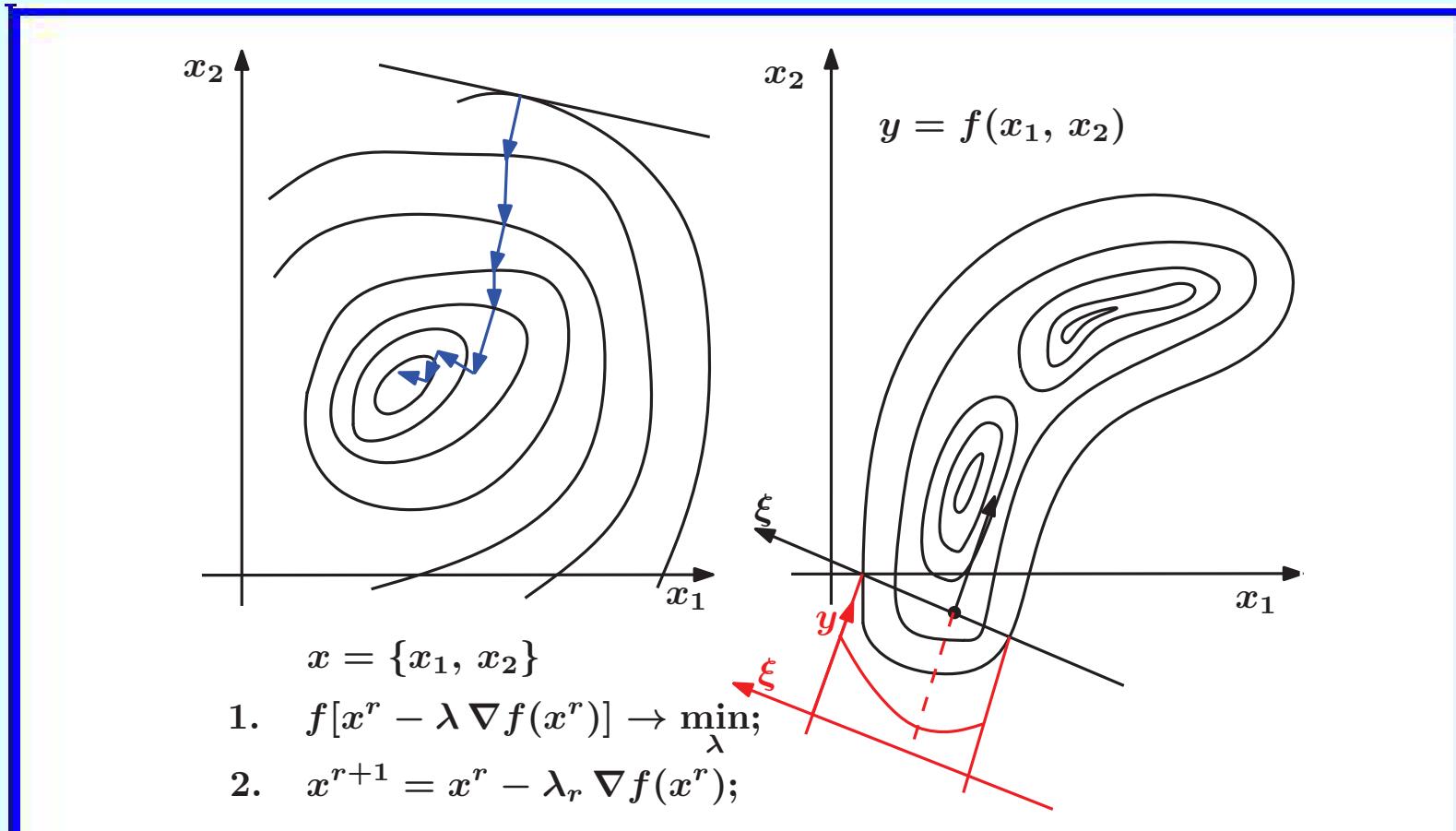
Обучение сетей прямого распространения (XXXV)

Рельеф функции ошибки



Обучение сетей прямого распространения (XXXVI)

Градиентный поиск минимума функции

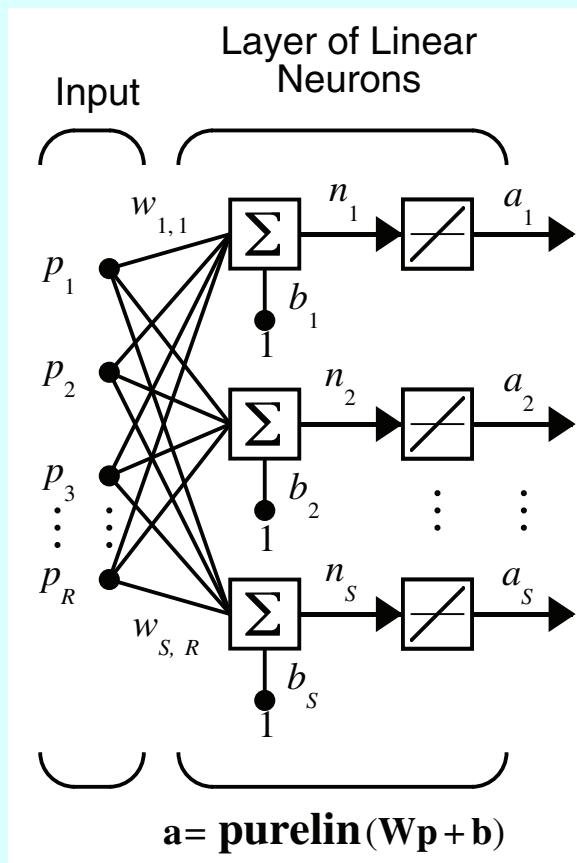


1. Общая схема градиентного поиска (слева).
2. Схема подзадачи одномерного поиска (справа).

Обучение сетей прямого распространения (XXXVII)

Обучение однослойной сети

Ошибка сети и дельта-правило



Дельта-правило:

$$w_{ij}^{r+1} = w_{ij}^r - \alpha \frac{\partial \mathcal{E}}{\partial w_{ij}^r},$$

$$b_i^{r+1} = b_i^r - \alpha \frac{\partial \mathcal{E}}{\partial b_i^r}$$

Производные ошибки сети \mathcal{E} по настраиваемым параметрам сети w_{ij} и b_i :

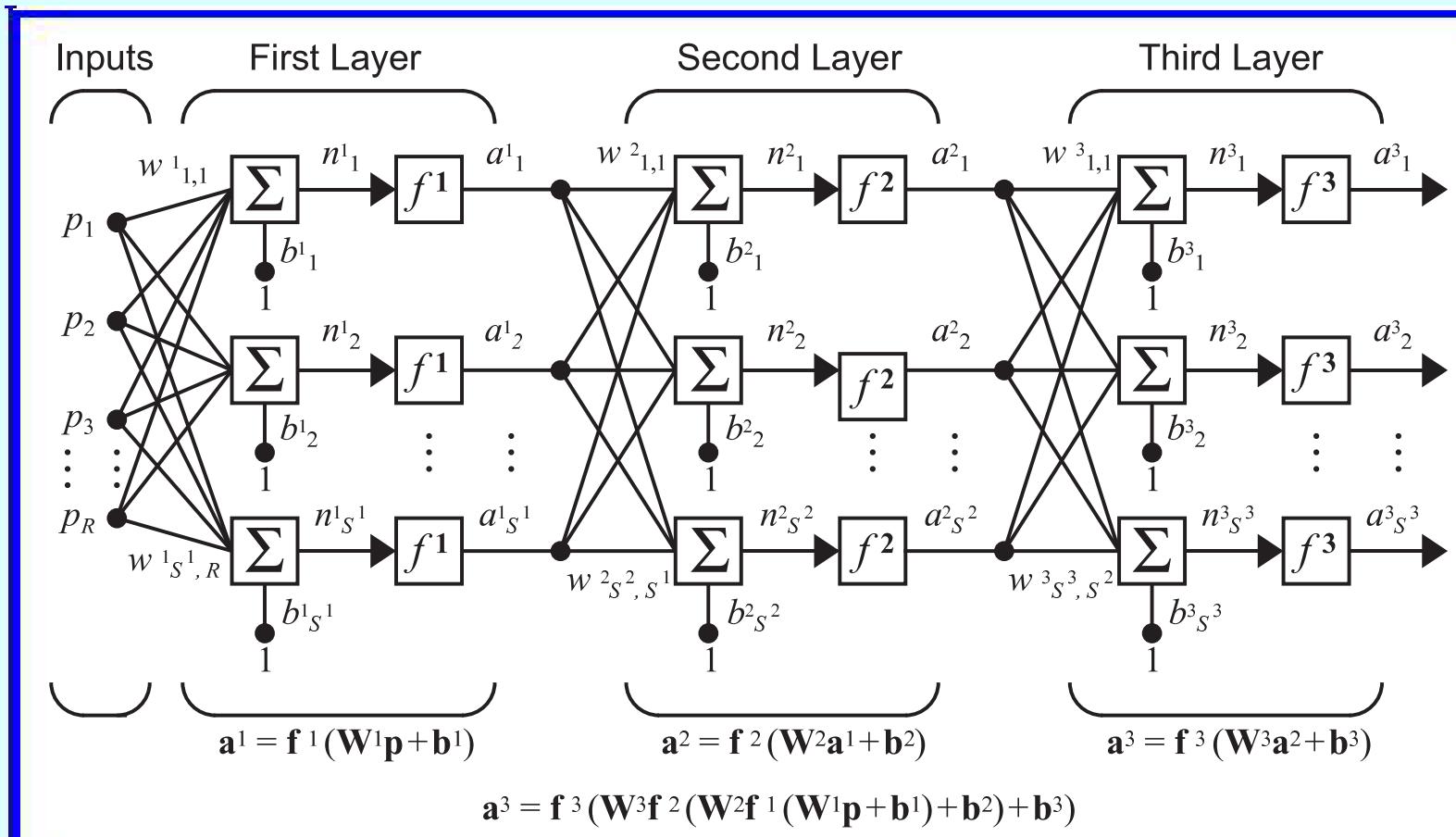
$$\frac{\partial \mathcal{E}}{\partial w_{ij}^r} = \frac{\partial \mathcal{E}}{\partial a_i^r} \frac{\partial a_i^r}{\partial w_{ij}^r} = -(t_i - a_i^r) p_j^r = -\mathcal{E}^r p_j$$

$$\frac{\partial \mathcal{E}}{\partial b_i^r} = \frac{\partial \mathcal{E}}{\partial a_i^r} \frac{\partial a_i^r}{\partial b_i^r} = -(t_i - a_i^r) = -\mathcal{E}^r$$

t_i — эталонный выход i -го нейрона
 $i = 1, \dots, S$ — номер нейрона
 $j = 1, \dots, R$ — номер входа
 $r = 1, 2, \dots$ — номер эпохи

Обучение сетей прямого распространения (XXXVIII)

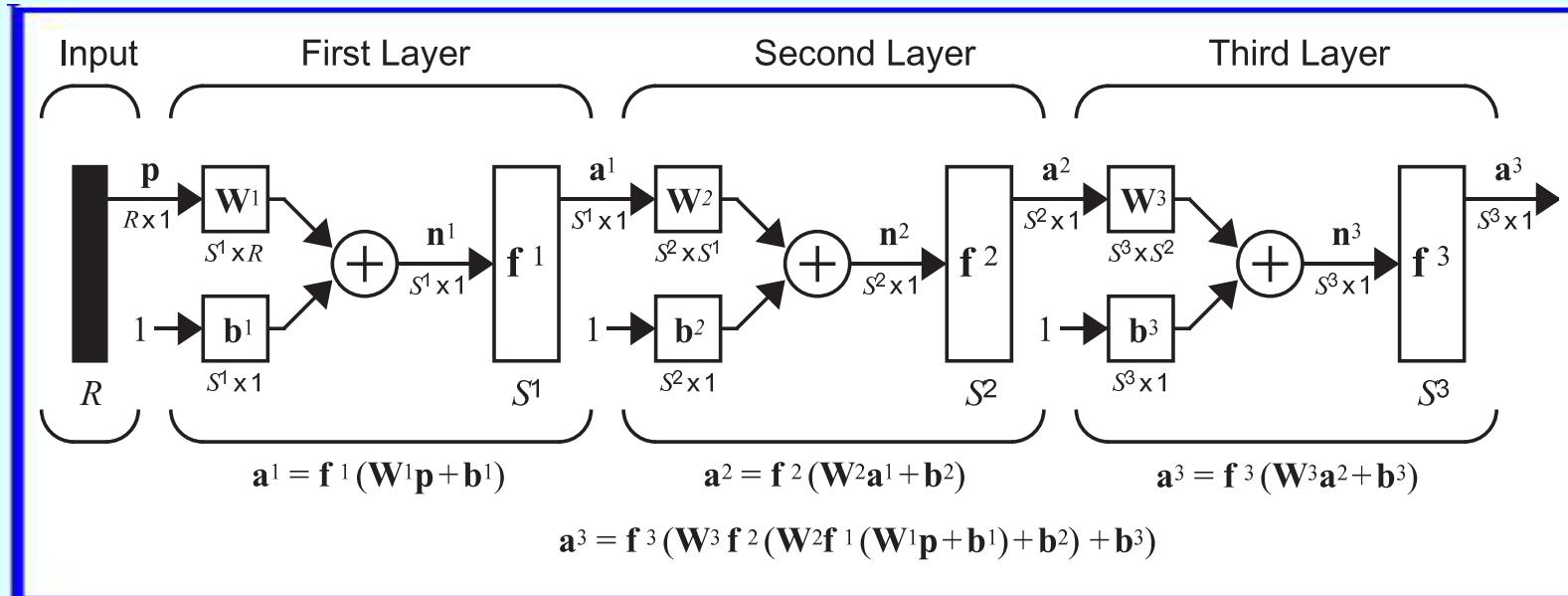
Обучение многослойной сети – 1



Проблема: Данные для непосредственного вычисления значений ошибки на выходах нейронов скрытых слоев **недоступны**.

Обучение сетей прямого распространения (XXXIX)

Обучение многослойной сети – 2



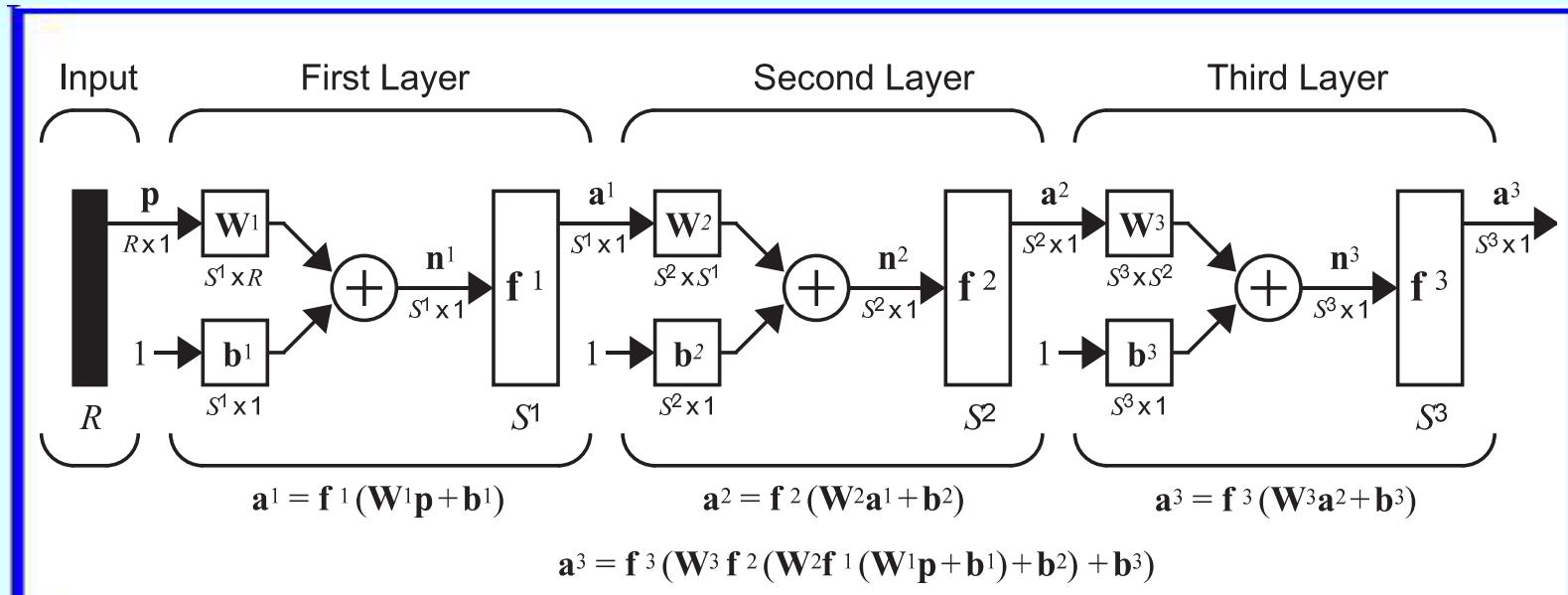
Ошибка сети $\mathcal{E}(\cdot)$ непосредственно связана лишь с выходами сети и весами связей выходного слоя и предшествующего ему скрытого слоя.

Надо уметь определять ошибку на выходах нейронов скрытых слоев по ошибке в последующем слое: ошибку для последнего скрытого слоя по ошибке выходного слоя, ошибку для предпоследнего скрытого слоя — по ошибке последнего скрытого слоя и т. д.

Ошибка всей сети «распространяется назад», от выходного слоя сети к ее входному слою, отсюда название соответствующего алгоритма: **алгоритм обратного распространения ошибки** (error backpropagation — BP).

Обучение сетей прямого распространения (XL)

Обучение многослойной сети – 3

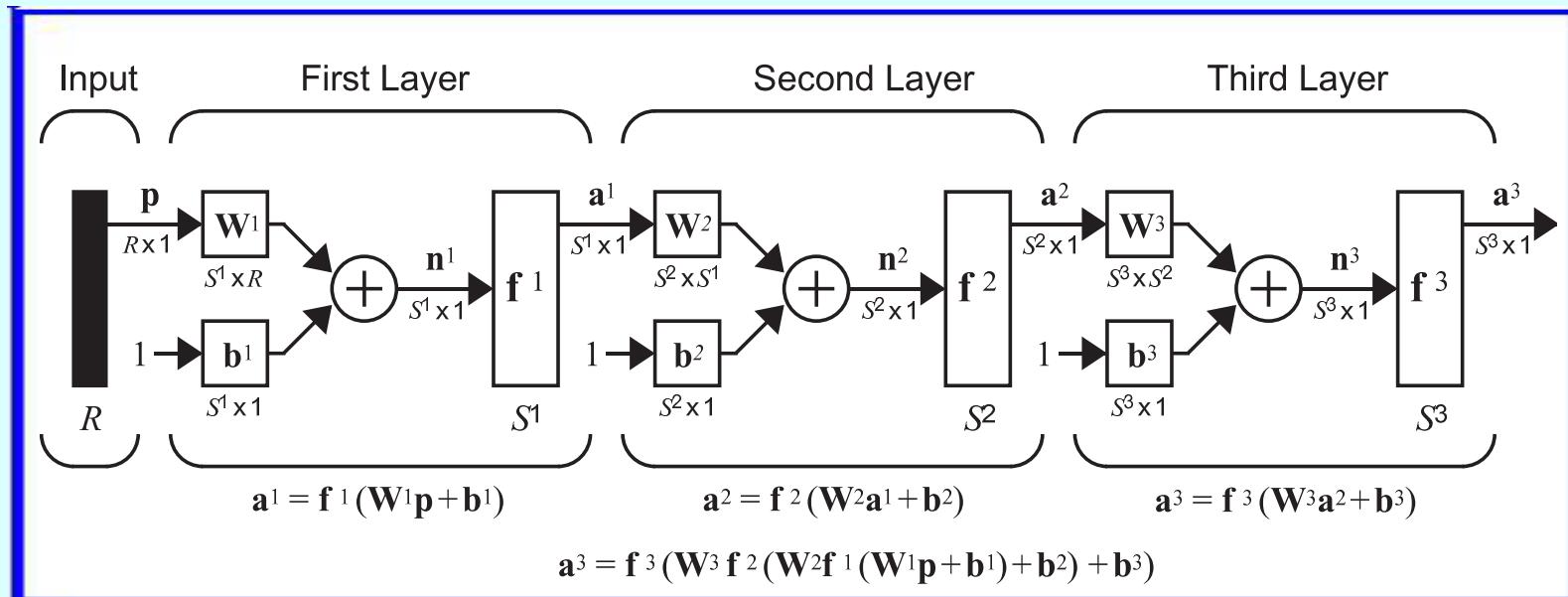


Переход «назад» от слоя к слою в алгоритме ВР сопровождается не только вычислением ошибки на соответствующем слое сети, но и **пересчетом весов** данного слоя, **уменьшающим его ошибку**.

Цель алгоритма ВР — при **вычислении градиента $\partial \mathcal{E} / \partial W$** уйти от **«проклятия размерности»**, связанного с порядком $\mathcal{O}(N_w^2)$ вычислительной сложности традиционных алгоритмов оптимизации.

Обучение сетей прямого распространения (XLI)

Обучение многослойной сети – 4



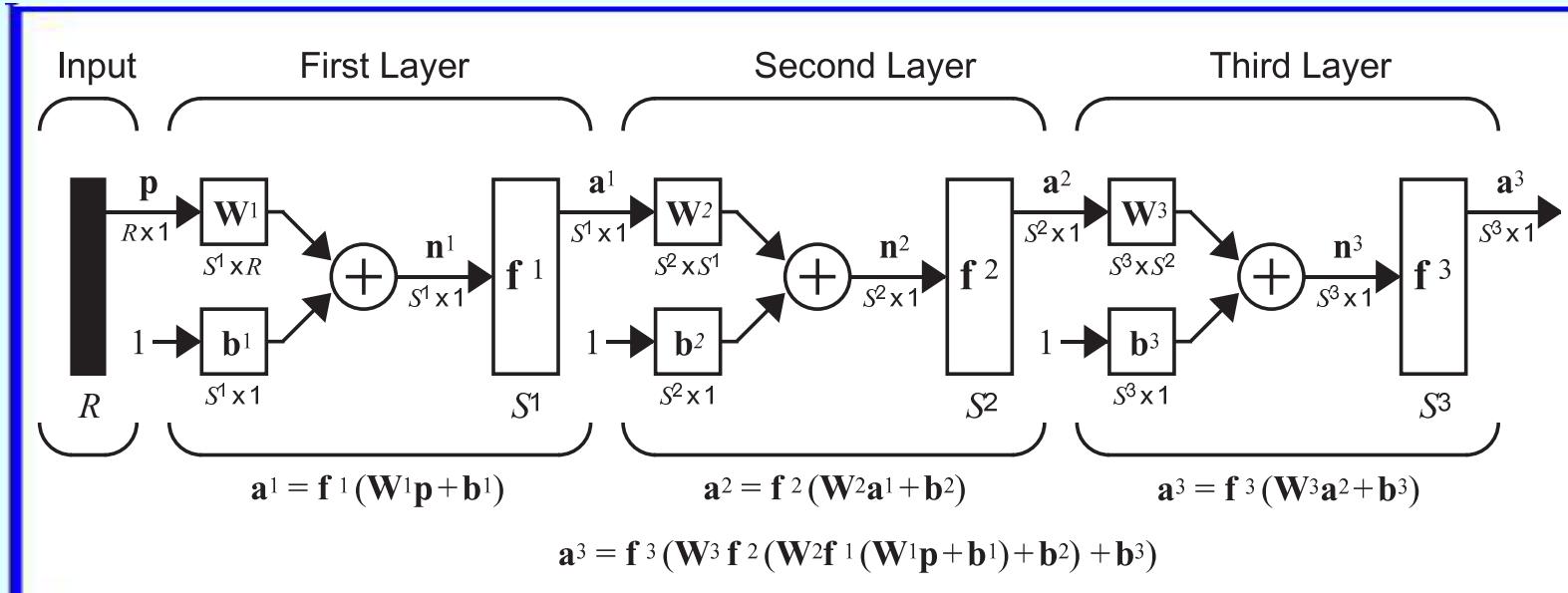
Выход слоя в многослойной сети:
 $(M$ – число слоев сети)

$$a^{m+1} = f^{m+1}(W^{m+1} a^m + b^{m+1}), \quad m = 0, 1, \dots, M - 1$$

$$a^0 = p, \quad a = a^M$$

Обучение сетей прямого распространения (XLII)

Обучение многослойной сети – 5



Среднеквадратическая ошибка многослойной сети:

$$E(\mathbf{W}) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k)) = \mathbf{e}^T(k) \mathbf{e}(k)$$

\mathbf{t} — эталоны выходов сети

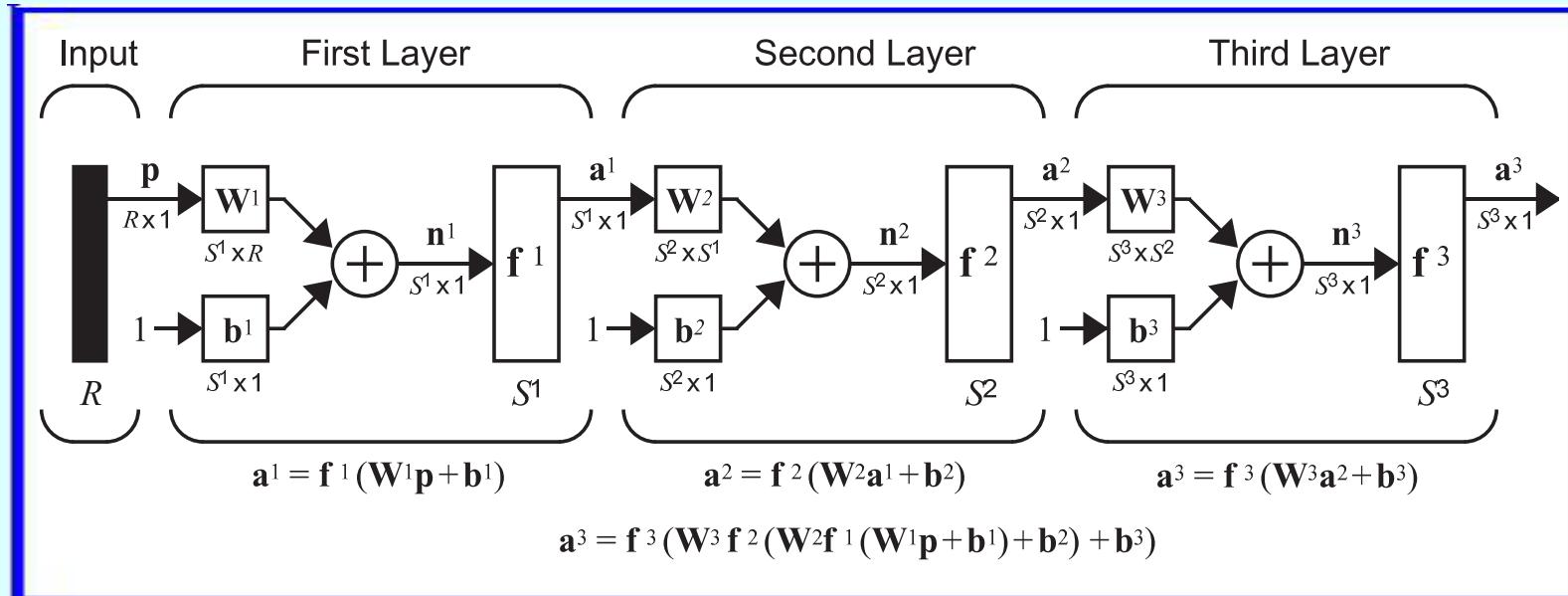
\mathbf{W} — веса сети

k — номер итерации

Алгоритм **BP** является развитием алгоритма **LMS**.
Оба алгоритма минимизируют ошибку сети, варьируя ее веса.

Обучение сетей прямого распространения (XLIII)

Обучение многослойной сети – 6



Алгоритм наискорейшего спуска для минимизации ошибки сети:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial E}{\partial w_{i,j}^m}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial E}{\partial b_i^m}$$

α — скорость обучения

Проблема: вычислить значения **частных производных** в выражениях для корректировки весов и смещений сети, обойти при этом «проклятие размерности».

Обучение сетей прямого распространения (XLIV)

Цепное правило – 1

Слоистая сеть представляет собой **композицию отображений** или, что то же самое, **сложную функцию**, например, если $y = f(u)$, где $u = \varphi(x)$, такая функция имеет вид $y = f(\varphi(x))$.

Производная сложной функции равна произведению ее производной по промежуточному аргументу на производную этого аргумента по независимой переменной (**цепное правило дифференцирования**):

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Применение этого правила играет **ключевую роль** в методе обратного распространения ошибки и его разновидностях.

Используя цепное правило дифференцирования, можно осуществить **вычисление градиента $\partial \mathcal{E} / \partial \mathbf{W}$** функции ошибки.

Следовательно, требуется **дифференцируемость активационных функций** в сети, а также желательность существования для этих функций производных в виде явных формул.

Обучение сетей прямого распространения (ХLV)

Цепное правило – 2

Производные функции ошибки по весам и смещениям сети:

$$\frac{\partial E}{\partial w_{i,j}^m} = \frac{\partial E}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m}$$

$$\frac{\partial E}{\partial b_i^m} = \frac{\partial E}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}$$

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m$$

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \quad \frac{\partial n_i^m}{\partial b_i^m} = 1$$

Здесь S^{m-1} — число нейронов в слое с номером $(m - 1)$.

$$\delta_i^m \equiv \frac{\partial E}{\partial n_i^m}$$

Здесь δ_i^m — чувствительность E к изменению выхода сумматора i -го элемента из слоя с номером m .

Величину δ_i^m именуют также невязкой i -го нейрона из m -го слоя, поскольку она характеризует вклад данного нейрона в значение ошибки сети E .

Обучение сетей прямого распространения (XLVI)

Цепное правило – 3

Производные функции ошибки по весам и смещениям сети:
(продолжение)

$$\frac{\partial E}{\partial w_{i,j}^m} = \delta_i^m a_j^{m-1}, \quad \frac{\partial E}{\partial b_i^m} = \delta_i^m$$

Алгоритм наискорейшего спуска для минимизации ошибки сети:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \delta_i^m a_j^{m-1}$$
$$b_i^m(k+1) = b_i^m(k) - \alpha \delta_i^m$$

Алгоритм наискорейшего спуска в матричной форме:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \delta^m (\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \delta^m$$

$$\delta^m \equiv \frac{\partial E}{\partial \mathbf{n}^m} = \left[\frac{\partial E}{\partial n_1^m} \ \frac{\partial E}{\partial n_2^m} \ \dots \ \frac{\partial E}{\partial n_{S^m}^m} \right]^T$$

Проблема: Найти значения невязок (чувствительностей) δ_i^m .

Обучение сетей прямого распространения (XLVII)

Цепное правило – 4

Сопоставление алгоритма наискорейшего спуска
для минимизации ошибки сети
с алгоритмом LMS

Алгоритм наискорейшего спуска:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \boldsymbol{\delta}^m (\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \boldsymbol{\delta}^m$$

Алгоритм LMS:

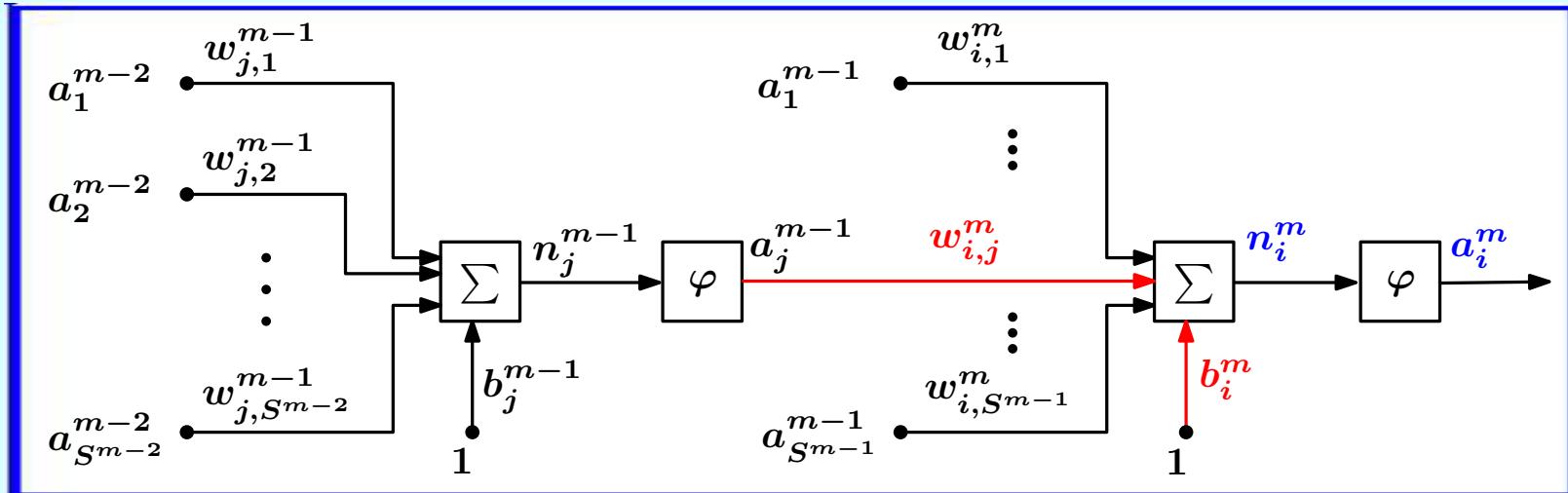
$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha \mathbf{e}(k) \mathbf{p}^T(k)$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha \mathbf{e}(k)$$

В алгоритме наискорейшего спуска **осталось получить соотношения для вычисления $\boldsymbol{\delta}^m$** , тогда алгоритм будет полностью определенным.

Обучение сетей прямого распространения (XLVIII)

Обучение многослойной сети – 5



Корректировка весов связей для m -го слоя сети:

$\mathbf{p} = (p_1, p_2, \dots, p_n)$ — входной вектор сети,

$\mathbf{a} = (a_1, a_2, \dots, a_q)$ — выходной вектор сети,

M — число слоев сети, S^m — число элементов в m -м слое сети

\mathbf{W}^m — матрица связей, которые идут к слою с номером m

$w_{i,j}^m$ — вес связи между j -м нейроном $(m - 1)$ -го слоя и i -м нейроном m -го слоя

b_i^m — смещение i -го нейрона m -го слоя

m — номер слоя, которому принадлежит нейрон; i — номер нейрона в слое m ;

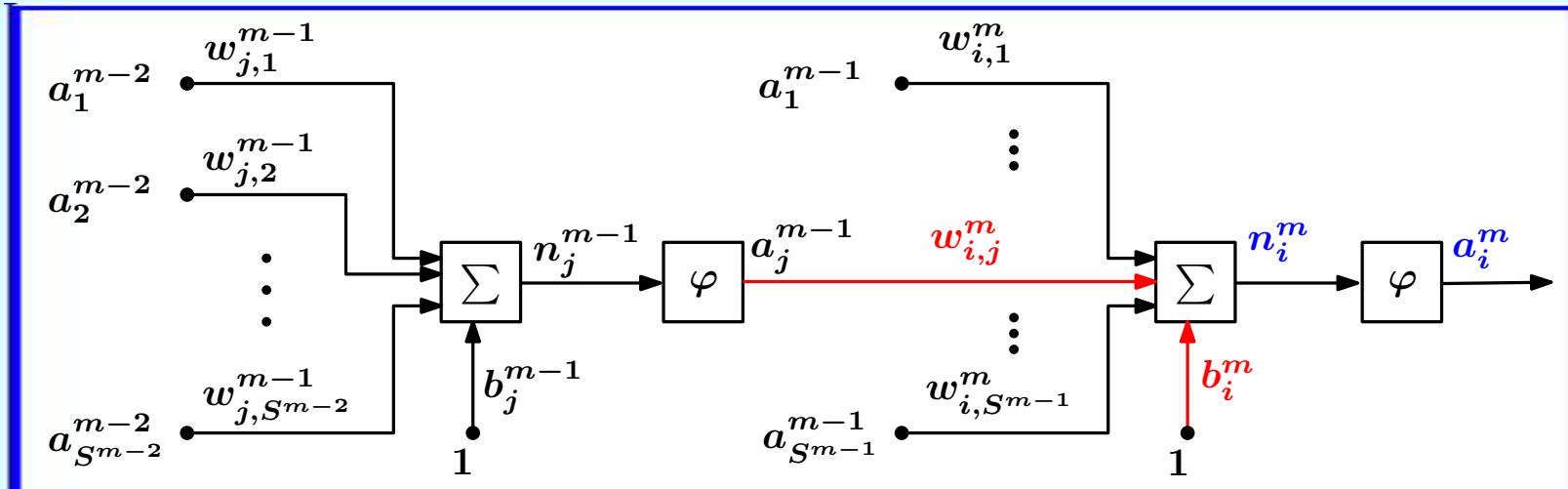
j — номер нейрона из предыдущего слоя, от которого пришла связь

$$a_1^0 = p_1, a_2^0 = p_2, \dots, a_n^0 = p_n$$

$$a_1^M = a_1, a_2^M = a_2, \dots, a_q^M = a_q$$

Обучение сетей прямого распространения (XLIX)

Обучение многослойной сети – 6



Корректировка весов связей для m -го слоя сети:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial E}{\partial w_{i,j}^m}, \quad i = 1, 2, \dots, S^m$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial E}{\partial b_i^m}$$

α — скорость обучения

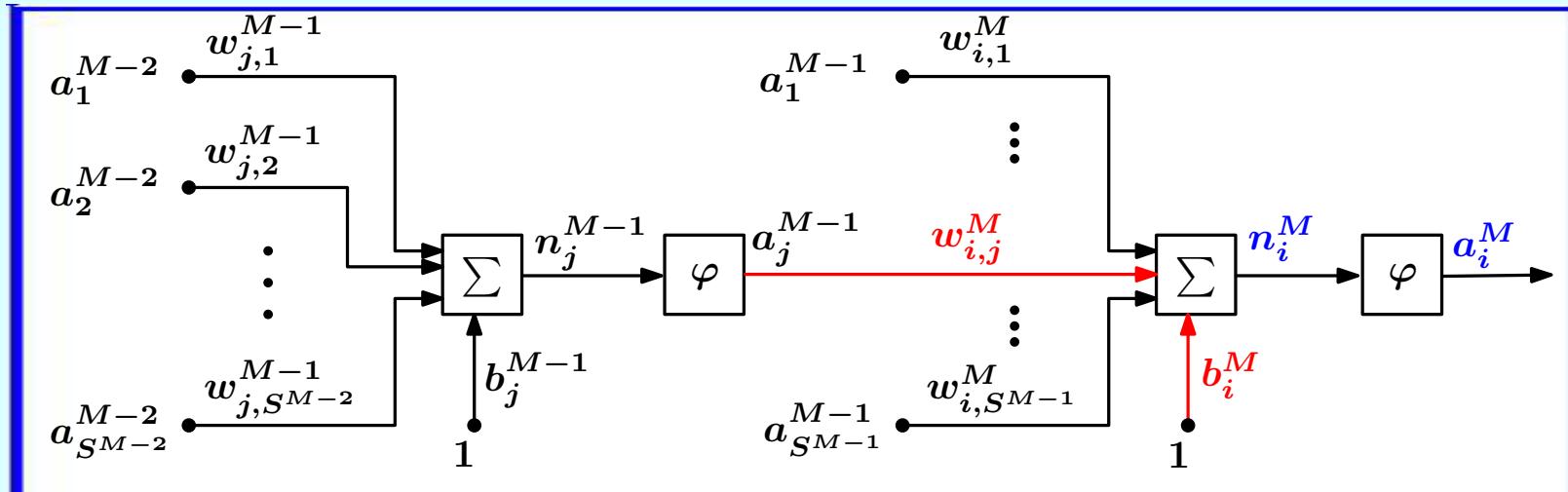
k — номер шага обучения

Проблема: найти значения частных производных:

$$\frac{\partial E}{\partial w_{i,j}^m}, \quad \frac{\partial E}{\partial b_i^m}$$

Обучение сетей прямого распространения (L)

Обучение многослойной сети – 7



Корректировка весов связей для выходного слоя сети:

$$w_{i,j}^M(k+1) = w_{i,j}^M(k) - \alpha \frac{\partial E}{\partial w_{i,j}^M}, \quad b_i^M(k+1) = b_i^M(k) - \alpha \frac{\partial E}{\partial b_i^M}$$

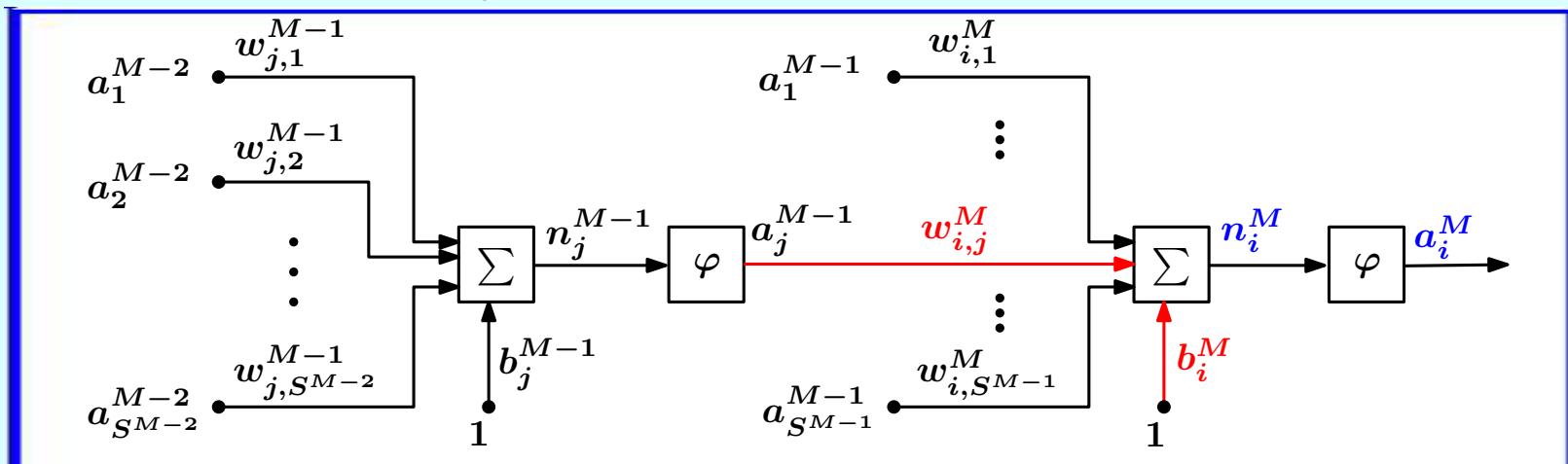
$$E = \frac{1}{2} \sum_{i=1}^{S^M} (t_i - a_i^M)^2, \quad a_i^M = \frac{1}{1 + e^{-n_i^M}}$$

$$n_i^M = \sum_{j=1}^{S^{M-1}} w_{i,j}^M a_j^{M-1} + b_i^M$$

$$\frac{\partial E}{\partial w_{i,j}^M} = \frac{\partial E}{\partial a_i^M} \cdot \frac{\partial a_i^M}{\partial n_i^M} \cdot \frac{\partial n_i^M}{\partial w_{i,j}^M}, \quad \frac{\partial E}{\partial b_i^M} = \frac{\partial E}{\partial a_i^M} \cdot \frac{\partial a_i^M}{\partial n_i^M}$$

Обучение сетей прямого распространения (LI)

Обучение многослойной сети – 8



Корректировка весов связей для выходного слоя сети:

$$\frac{\partial E}{\partial w_{i,j}^M} = \frac{\partial E}{\partial a_i^M} \cdot \frac{\partial a_i^M}{\partial n_i^M} \cdot \frac{\partial n_i^M}{\partial w_{i,j}^M}, \quad \frac{\partial E}{\partial b_i^M} = \frac{\partial E}{\partial a_i^M} \cdot \frac{\partial a_i^M}{\partial n_i^M}$$

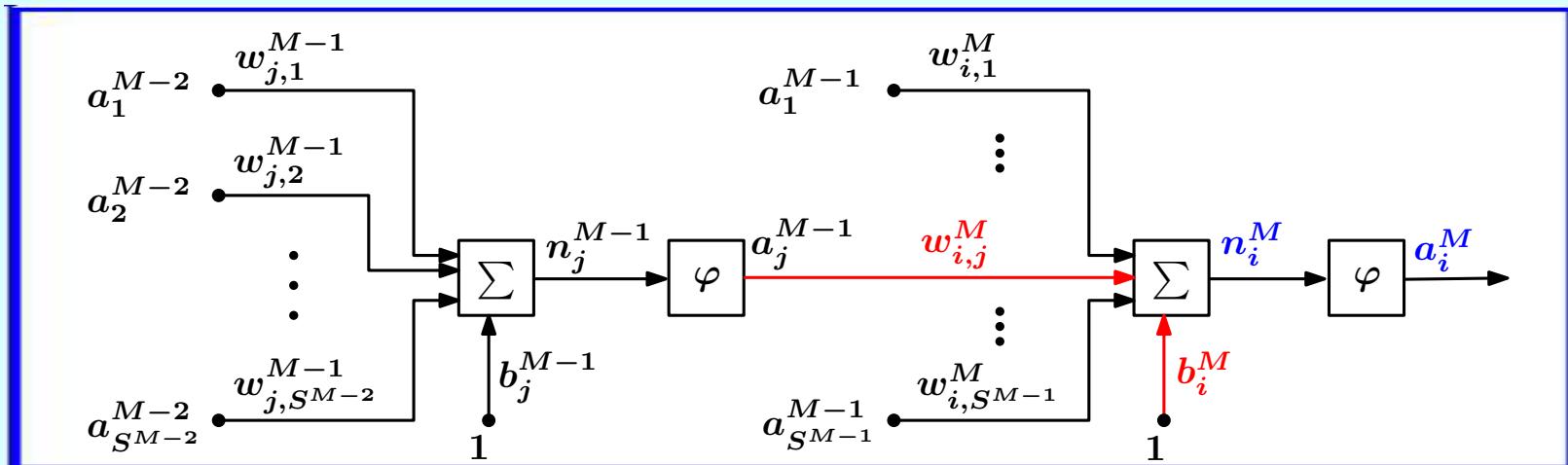
$$E = \frac{1}{2} \sum_{i=1}^{S^M} (t_i - a_i^M)^2 \Rightarrow \frac{\partial E}{\partial a_i^M} = -(t_i - a_i^M)$$

$$a_i^M = \frac{1}{1 + e^{-n_i^M}} \Rightarrow \frac{\partial a_i^M}{\partial n_i^M} = \left(\frac{1}{1 + e^{-n_i^M}} \right) \left(1 - \frac{1}{1 + e^{-n_i^M}} \right) = a_i^M (1 - a_i^M)$$

$$n_i^M = \sum_{j=1}^{S^{M-1}} w_{i,j}^M a_j^{M-1} + b_i^M \Rightarrow \frac{\partial n_i^M}{\partial w_{i,j}^M} = a_j^{M-1}$$

Обучение сетей прямого распространения (ЛII)

Обучение многослойной сети – 9



Корректировка весов связей для выходного слоя сети:

$$\frac{\partial E}{\partial w_{i,j}^M} = \frac{\partial E}{\partial a_i^M} \cdot \frac{\partial a_i^M}{\partial n_i^M} \cdot \frac{\partial n_i^M}{\partial w_{i,j}^M}, \quad \frac{\partial E}{\partial b_i^M} = \frac{\partial E}{\partial a_i^M} \cdot \frac{\partial a_i^M}{\partial n_i^M}$$

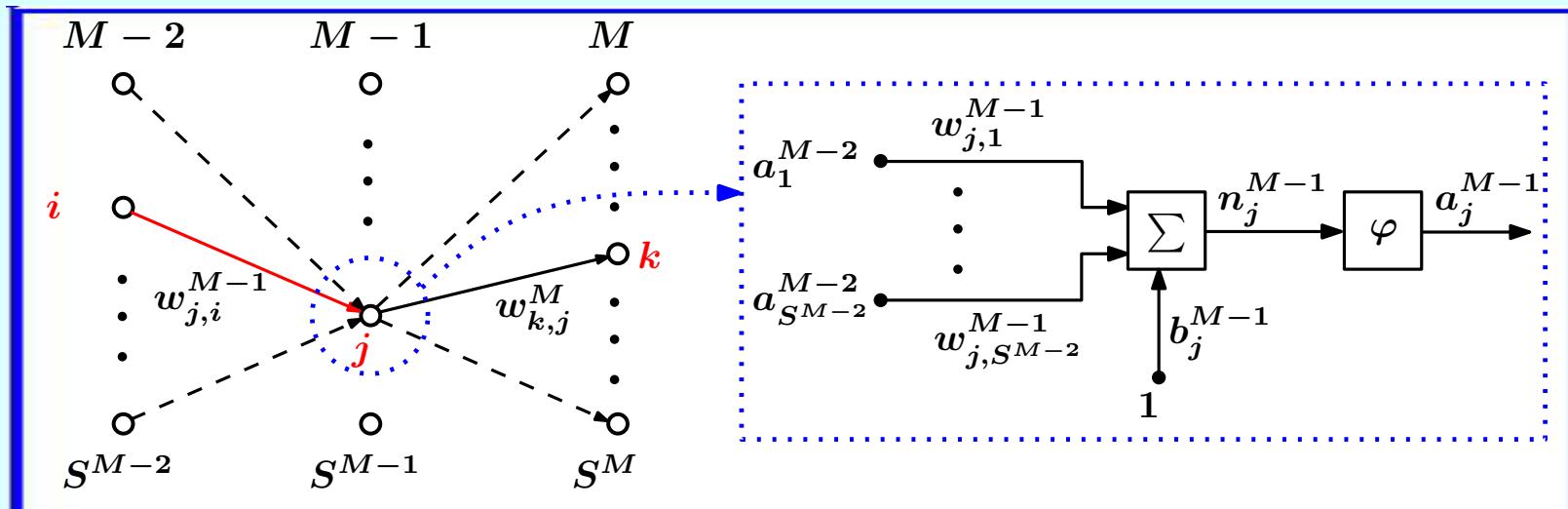
$$\frac{\partial E}{\partial w_{i,j}^M} = -(t_i - a_i^M) a_i^M (1 - a_i^M) a_j^{M-1}, \quad \frac{\partial E}{\partial b_i^M} = -(t_i - a_i^M) a_i^M (1 - a_i^M)$$

$$\delta_i^M = -(t_i - a_i^M) a_i^M (1 - a_i^M) \Rightarrow \Delta w_{i,j}^M = \alpha \delta_i^M a_i^M, \quad \Delta b_i^M = \alpha \delta_i^M$$

δ_i^M – невязка i -го нейрона выходного (M -го) слоя сети.

Обучение сетей прямого распространения (ЛIII)

Обучение многослойной сети – 10



Корректировка весов связей для скрытого слоя сети:

$$w_{j,i}^{M-1}(r+1) = w_{j,i}^{M-1}(r) - \alpha \frac{\partial E}{\partial w_{j,i}^{M-1}}, \quad j = 1, 2, \dots, S^{M-1}$$

$$b_i^{M-1}(r+1) = b_i^{M-1}(r) - \alpha \frac{\partial E}{\partial b_i^{M-1}}$$

α — скорость обучения

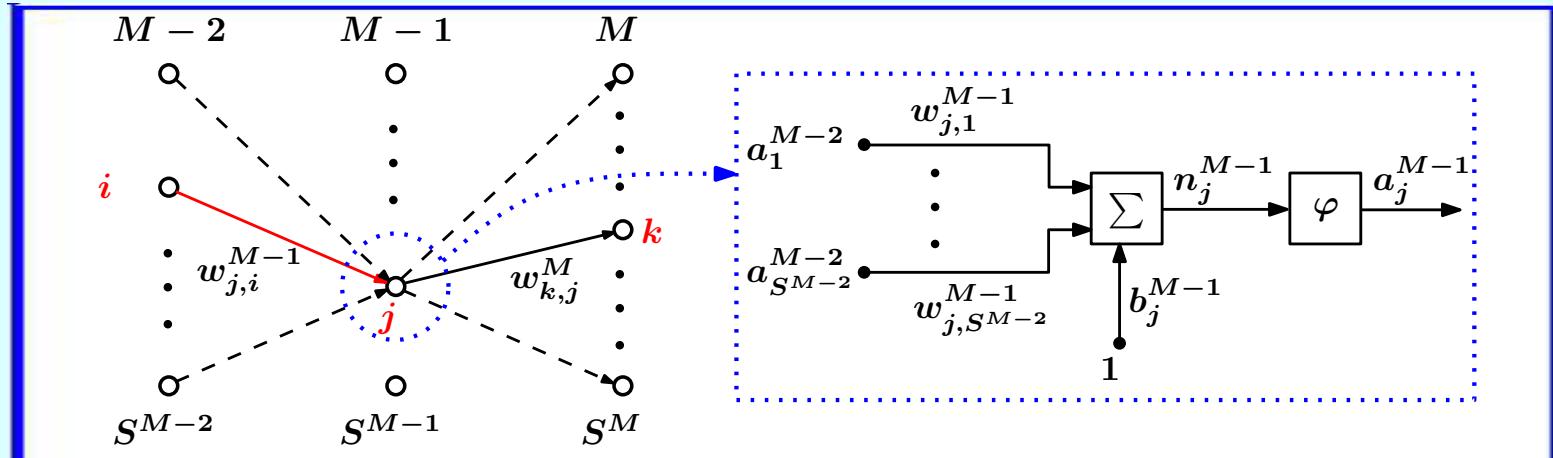
r — номер шага обучения

Цепочка изменений:

$$w_{j,i}^{M-1} \Rightarrow n_i^{M-1} \Rightarrow a_i^{M-1} \Rightarrow n_k^M \Rightarrow a_k^M, \forall k \in \{1, \dots, S^M\}$$

Обучение сетей прямого распространения (Л4)

Обучение многослойной сети – 11



$$E = \frac{1}{2} \sum_{i=k}^{S^M} (t_k - a_k^M)^2$$

$$\downarrow$$

$$a_k^M = \frac{1}{1 + e^{-n_k^M}} \Rightarrow n_k^M = \sum_{j=1}^{S^{M-1}} w_{k,j}^M a_j^{M-1} + b_k^M$$

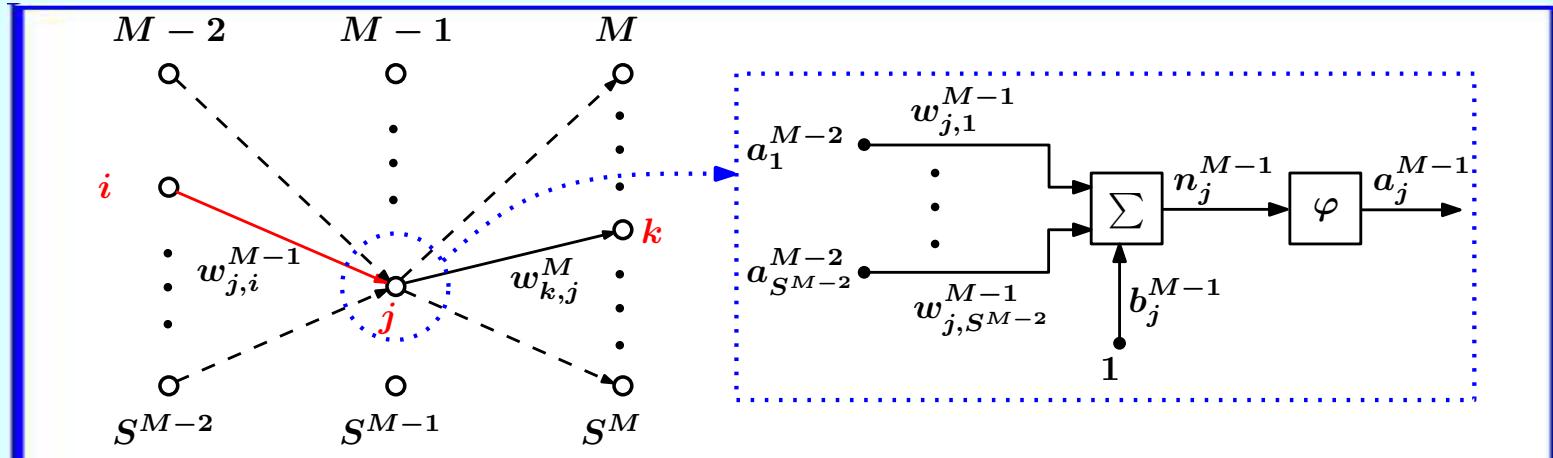
$$\downarrow$$

$$a_j^{M-1} = \frac{1}{1 + e^{-n_j^{M-1}}} \Rightarrow n_j^{M-1} = \sum_{i=1}^{S^{M-2}} w_{j,i}^{M-1} a_i^{M-2} + b_j^{M-1}$$

$$\frac{\partial E}{\partial w_{j,i}^{M-1}} = \sum_{k=1}^{S^M} \frac{\partial E}{\partial a_k^M} \cdot \frac{\partial a_k^M}{\partial n_k^M} \cdot \frac{\partial n_k^M}{\partial a_j^{M-1}} \cdot \frac{\partial a_j^{M-1}}{\partial n_j^{M-1}} \cdot \frac{\partial n_j^{M-1}}{\partial w_{j,i}^{M-1}}$$

Обучение сетей прямого распространения (LV)

Обучение многослойной сети – 12



$$\frac{\partial E}{\partial w_{j,i}^{M-1}} = \sum_{k=1}^{S^M} \frac{\partial E}{\partial a_k^M} \cdot \frac{\partial a_k^M}{\partial n_k^M} \cdot \frac{\partial n_k^M}{\partial a_j^{M-1}} \cdot \frac{\partial a_j^{M-1}}{\partial n_j^{M-1}} \cdot \frac{\partial n_j^{M-1}}{\partial w_{j,i}^{M-1}}$$

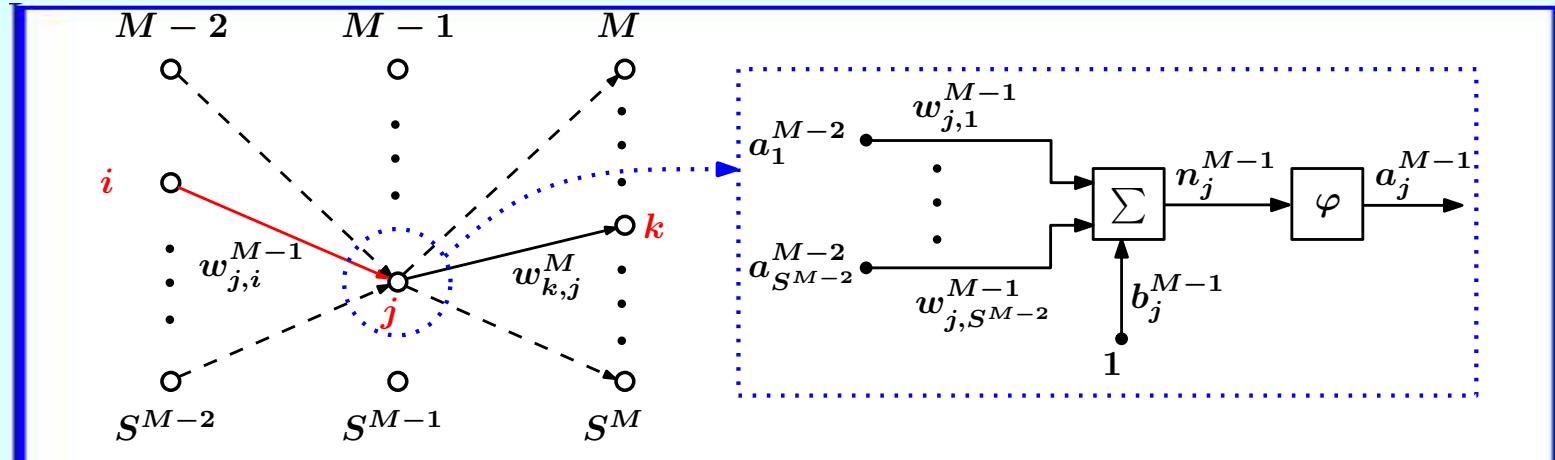
$$\frac{\partial E}{\partial a_k^M} = -(t_k - a_k^M), \quad \frac{\partial a_k^M}{\partial n_k^M} = a_k^M (1 - a_k^M), \quad \frac{\partial n_k^M}{\partial a_j^{M-1}} = w_{k,j}^M$$

$$\frac{\partial a_j^{M-1}}{\partial n_j^{M-1}} = a_j^{M-1} (1 - a_j^{M-1}), \quad \frac{\partial n_j^{M-1}}{\partial w_{i,j}^M} = a_i^{M-2}$$

$$\frac{\partial E}{\partial w_{j,i}^{M-1}} = \sum_{k=1}^{S^M} -(t_k - a_k^M) a_k^M (1 - a_k^M) w_{k,j}^M a_j^{M-1} (1 - a_j^{M-1}) a_i^{M-2}$$

Обучение сетей прямого распространения (LVI)

Обучение многослойной сети – 13



$$\frac{\partial E}{\partial w_{j,i}^{M-1}} = \sum_{k=1}^{S^M} -(t_k - a_k^M) a_k^M (1 - a_k^M) w_{k,j}^M a_j^{M-1} (1 - a_j^{M-1}) a_i^{M-2}$$

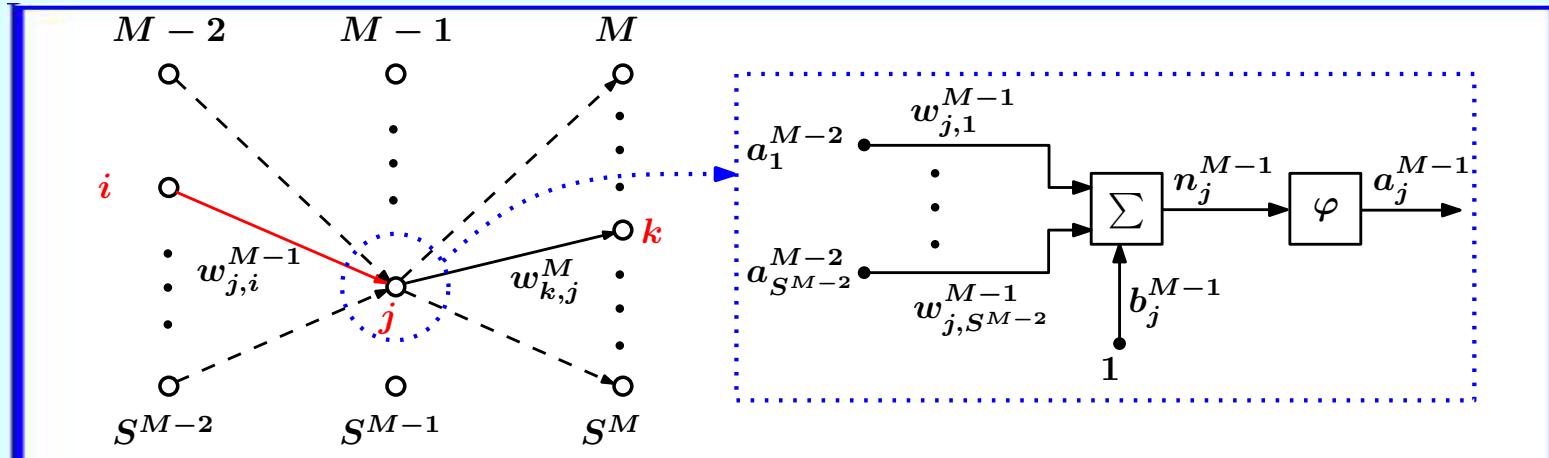
$$\delta_k^M = (t_k - a_k^M) a_k^M (1 - a_k^M)$$

$$\begin{aligned} \frac{\partial E}{\partial w_{j,i}^{M-1}} &= \sum_{k=1}^{S^M} -\delta_k^M w_{k,j}^M a_j^{M-1} (1 - a_j^{M-1}) a_i^{M-2} = \\ &= -a_i^{M-2} a_j^{M-1} (1 - a_j^{M-1}) \sum_{k=1}^{S^M} \delta_k^M w_{k,j}^M \end{aligned}$$

$$\delta_j^{M-1} = a_j^{M-1} (1 - a_j^{M-1}) \sum_{k=1}^{S^M} \delta_k^M w_{k,j}^M \Rightarrow \Delta w_{j,i}^{M-1} = \alpha \delta_j^{M-1} a_i^{M-2}$$

Обучение сетей прямого распространения (LVII)

Обучение многослойной сети – 14



$$\frac{\partial E}{\partial w_{j,i}^{M-1}} = -a_i^{M-2} a_j^{M-1} (1 - a_j^{M-1}) \sum_{k=1}^{S^M} \delta_k^M w_{k,j}^M = -\delta_j^{M-1} a_i^{M-2}$$

$$\delta_j^{M-1} = a_j^{M-1} (1 - a_j^{M-1}) \sum_{k=1}^{S^M} \delta_k^M w_{k,j}^M$$

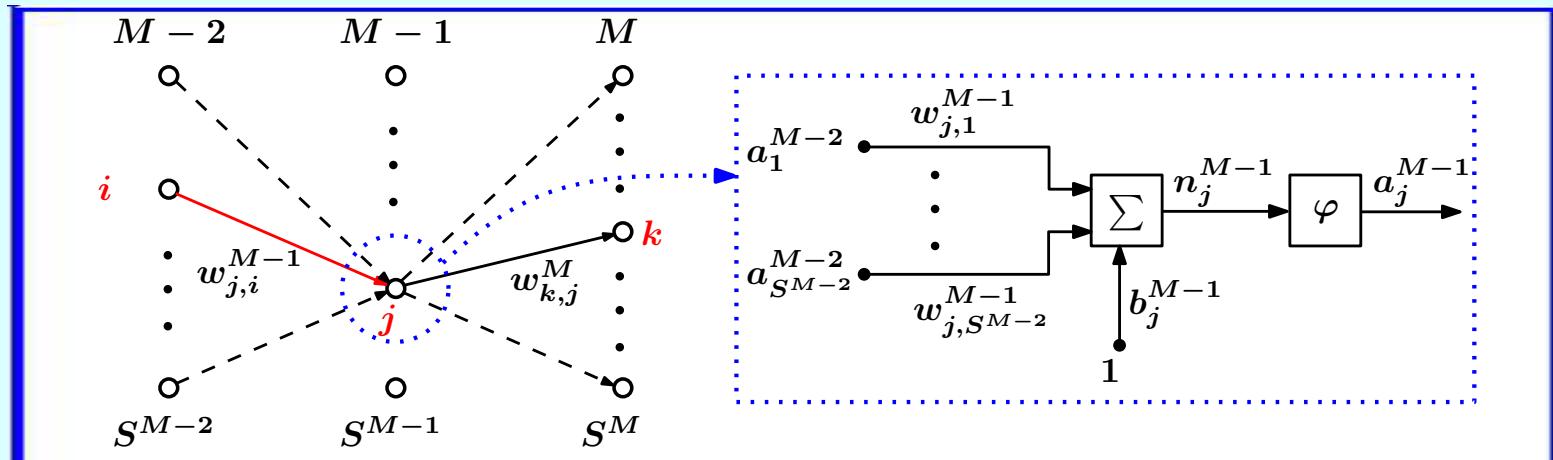
$$\delta_k^M = (t_k - a_k^M) a_k^M (1 - a_k^M)$$

Обратное распространение ошибки:

$$(t_k - a_k^M) \Rightarrow \delta_k^M \Rightarrow \delta_j^{M-1}$$

Обучение сетей прямого распространения (LVIII)

Обучение многослойной сети – 15



$$\frac{\partial E}{\partial w_{j,i}^{M-1}} = -a_i^{M-2} a_j^{M-1} (1 - a_j^{M-1}) \sum_{k=1}^{S^M} \delta_k^M w_{k,j}^M = -\delta_j^{M-1} a_i^{M-2}$$

$$\delta_j^{M-1} = a_j^{M-1} (1 - a_j^{M-1}) \sum_{k=1}^{S^M} \delta_k^M w_{k,j}^M$$

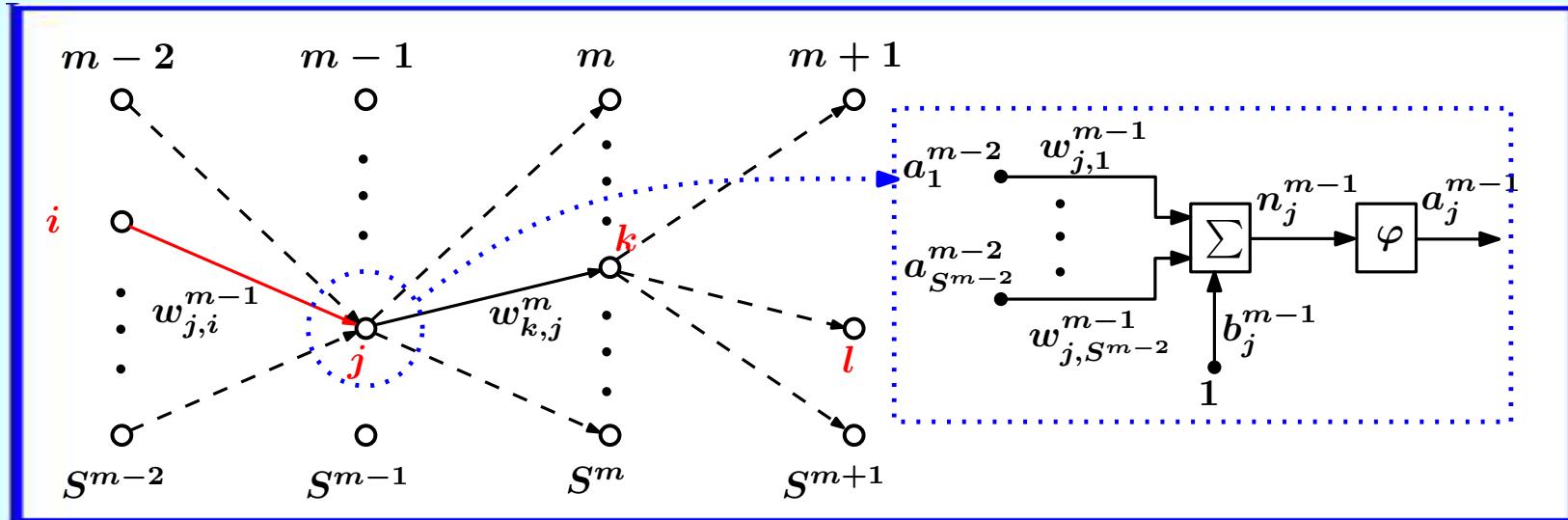
$$\delta_k^M = (t_k - a_k^M) a_k^M (1 - a_k^M)$$

Итак, **вклад в общую ошибку** для каждого веса может быть вычислен **локально**, простым умножением **невязки** нейрона $\delta_j^{(M-1)}$ на значение соответствующего **входа** a_i^{M-2} .

Поэтому **оценка вычислительной сложности** метода обратного распространения ошибки составляет $\mathcal{O}(N_W)$ вместо $\mathcal{O}(N_W^2)$ для традиционных методов оптимизации.

Обучение сетей прямого распространения (LIX)

Обучение многослойной сети – 16



**Корректировка весов произвольного скрытого слоя сети
(для j -го элемента из $(m-1)$ -го слоя сети)**

$$\frac{\partial E}{\partial w_{j,i}^{m-1}} = -a_i^{m-2} a_j^{m-1} (1 - a_j^{m-1}) \sum_{k=1}^{S^m} \delta_k^m w_{k,j}^m = -\delta_j^{m-1} a_i^{m-2}$$

$$\delta_j^{m-1} = a_j^{m-1} (1 - a_j^{m-1}) \sum_{k=1}^{S^m} \delta_k^m w_{k,j}^m$$

δ_k^m , $k = 1, \dots, S^m$ – с предыдущего шага обратного распространения ошибки,
на котором вычислялись невязки для элементов m -го слоя

Обучение сетей прямого распространения (ЛХ)

Алгоритм обратного распространения и алгоритм LMS

Алгоритм обратного распространения (BP):

$$\begin{aligned} \mathbf{W}^m(k+1) &= \mathbf{W}^m(k) - \alpha \delta^m (\mathbf{a}^{m-1})^T \\ \mathbf{b}^m(k+1) &= \mathbf{b}^m(k) - \alpha \delta^m \end{aligned}$$

Случай однослойной линейной сети (ADALINE):

$$\frac{\partial E}{\partial w_{i,j}^1} = \frac{\partial E}{\partial a_i^1} \cdot \frac{\partial a_i^1}{\partial n_i^1} \cdot \frac{\partial n_i^1}{\partial w_{i,j}^1}, \quad \frac{\partial E}{\partial b_i^1} = \frac{\partial E}{\partial a_i^1} \cdot \frac{\partial a_i^1}{\partial n_i^1}$$

$$E = \sum_i (t_i - a_i^1)^2, \quad a_i^1 = n_i^1, \quad n_i^1 = \sum_j w_{i,j}^1 a_j^0 + b_i^1, \quad a_j^0 = p_j$$

$$\frac{\partial E}{\partial a_i^1} = -2(t_i - a_i^1) = -2e_i, \quad \frac{\partial a_i^1}{\partial n_i^1} = 1, \quad \frac{\partial n_i^1}{\partial w_{i,j}^1} = a_j^0$$

$$\frac{\partial E}{\partial w_{i,j}^1} = -2(t_i - a_i^1)p_j = -2e_i p_j, \quad \frac{\partial E}{\partial b_i^1} = -2e_i, \quad \delta_i^1 = -2e_i$$

$$\mathbf{W}^1(k+1) = \mathbf{W}^1(k) - \alpha \delta^1 (\mathbf{a}^0)^T = \mathbf{W}^1(k) + 2\alpha e \mathbf{p}^T$$

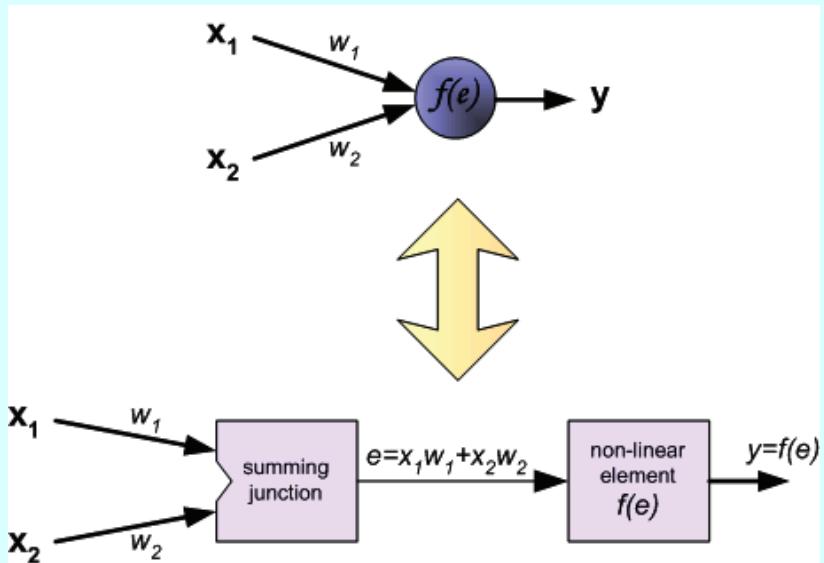
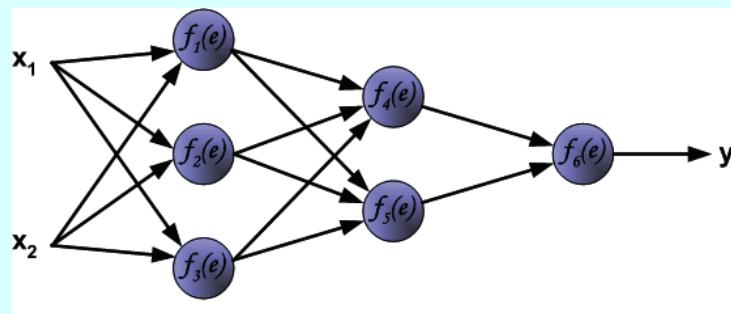
$$\mathbf{b}^1(k+1) = \mathbf{b}^1(k) - \alpha \delta^1 = \mathbf{b}^1(k) + 2\alpha e$$

Это в точности алгоритм LMS для однослойной линейной сети.
Таким образом, алгоритм BP можно считать обобщением алгоритма LMS.

Слоистые сети прямого распространения (I)

Пример работы алгоритма обратного распространения ошибки – 1

Структура модельной сети

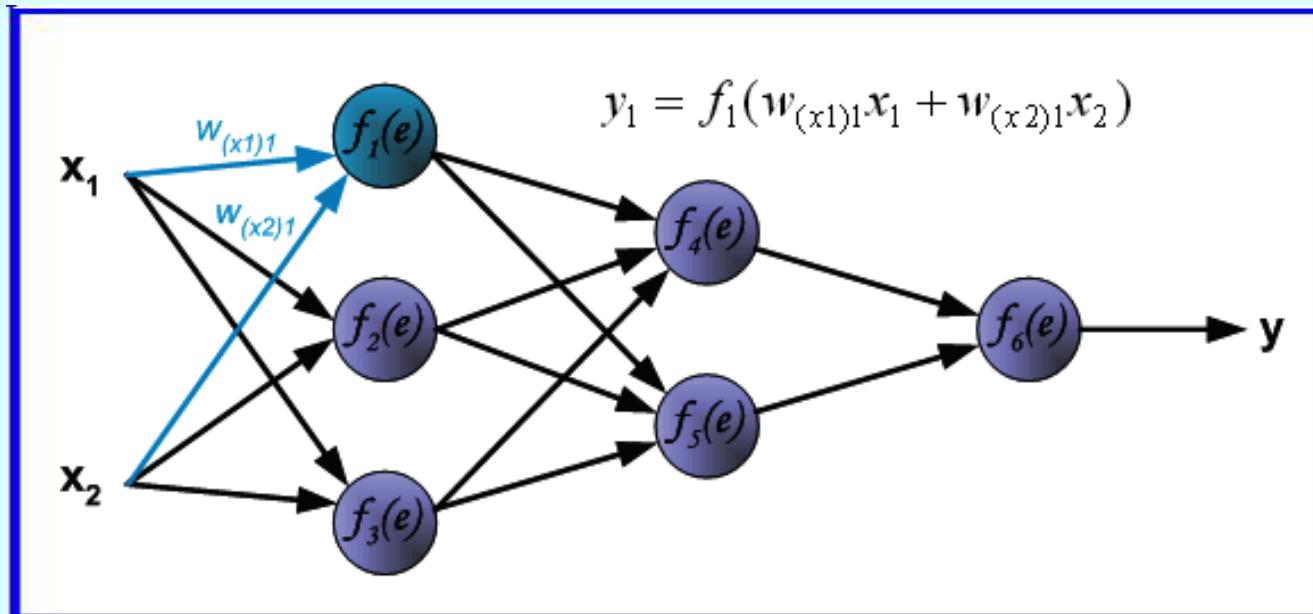


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Figs 1 and 1b)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (II)

Пример работы алгоритма обратного распространения ошибки – 2

Прохождение сигнала через сеть в прямом направлении (1)

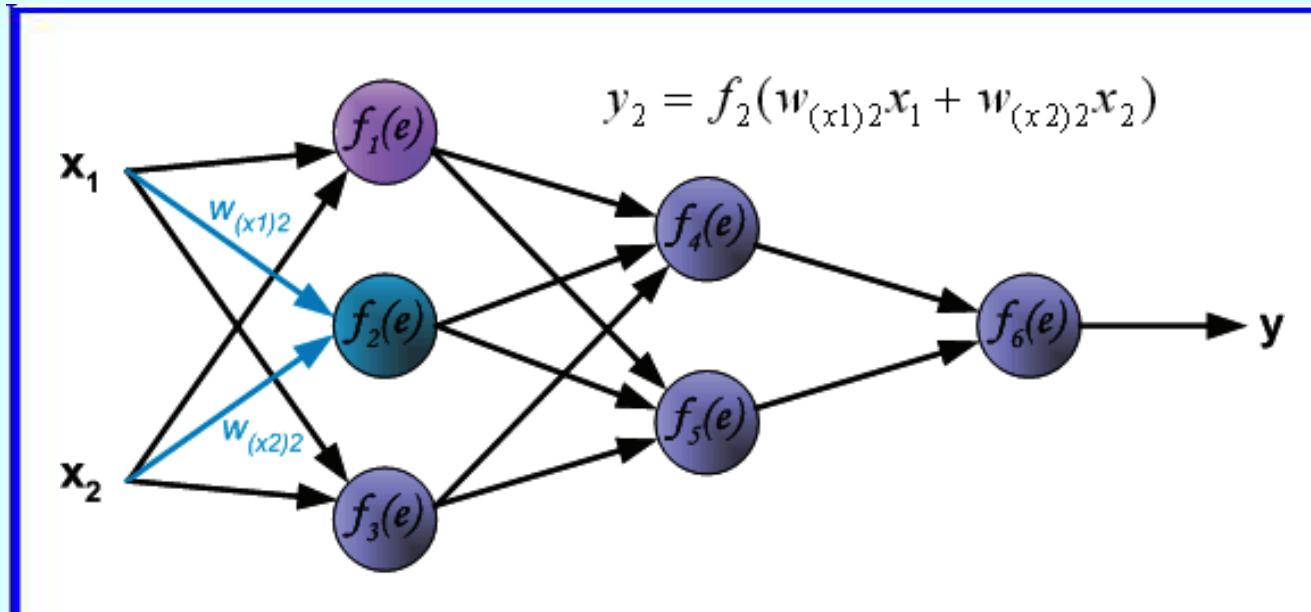


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 2)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (III)

Пример работы алгоритма обратного распространения ошибки – 3

Прохождение сигнала через сеть в прямом направлении (2)

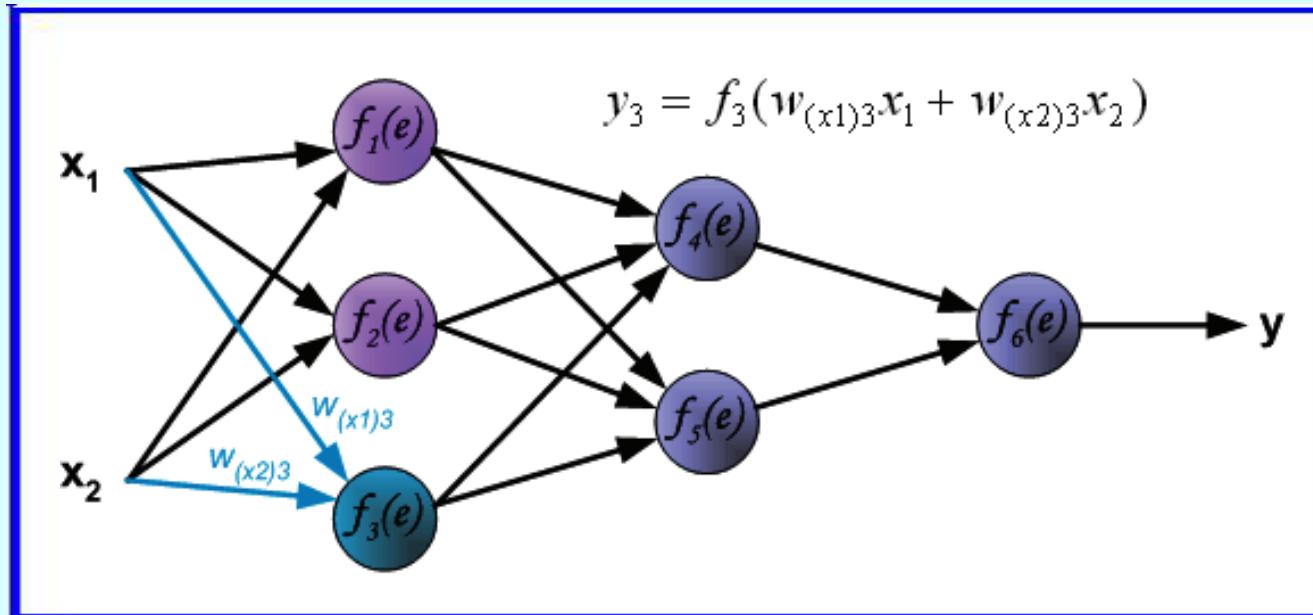


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 3)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (IV)

Пример работы алгоритма обратного распространения ошибки – 4

Прохождение сигнала через сеть в прямом направлении (3)

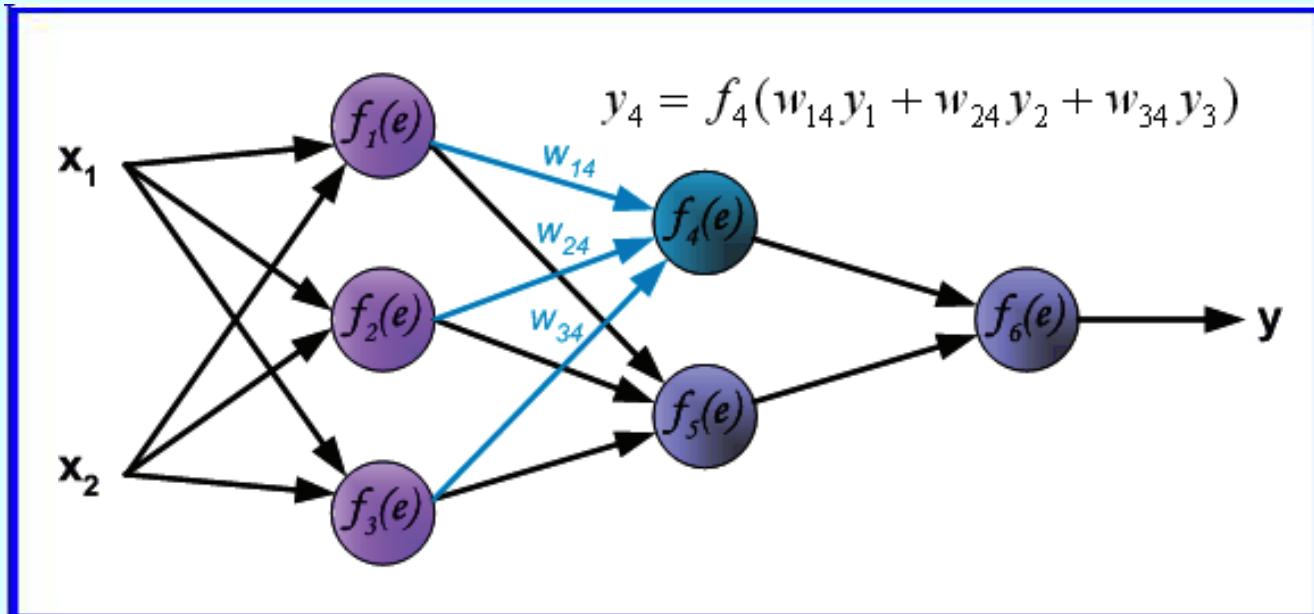


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 4)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (V)

Пример работы алгоритма обратного распространения ошибки – 5

Прохождение сигнала через сеть в прямом направлении (4)

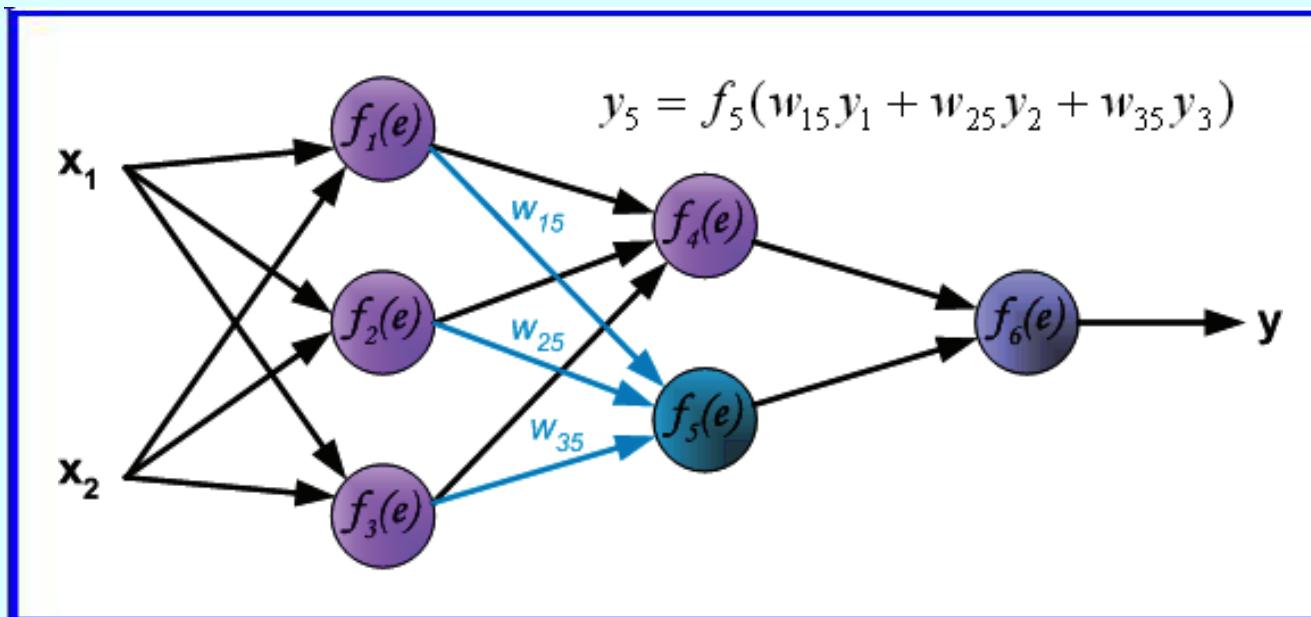


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 5)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (VI)

Пример работы алгоритма обратного распространения ошибки – 6

Прохождение сигнала через сеть в прямом направлении (5)

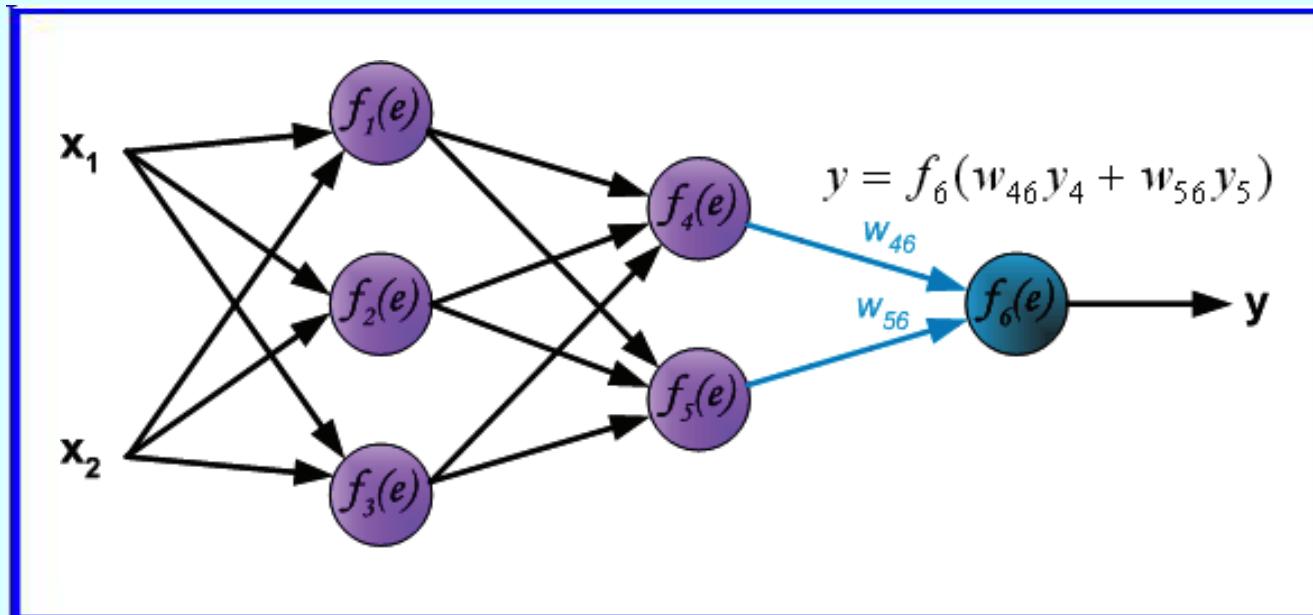


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 6)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (VII)

Пример работы алгоритма обратного распространения ошибки – 7

Прохождение сигнала через сеть в прямом направлении (6)

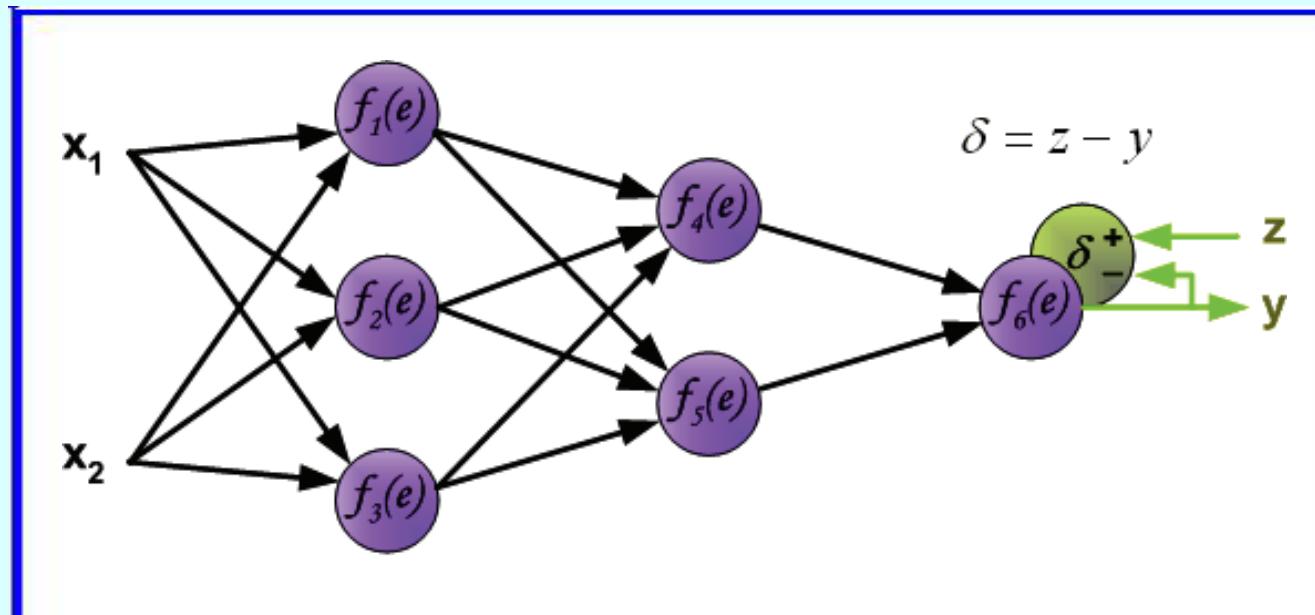


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 7)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (VIII)

Пример работы алгоритма обратного распространения ошибки – 8

Вычисление ошибки сети

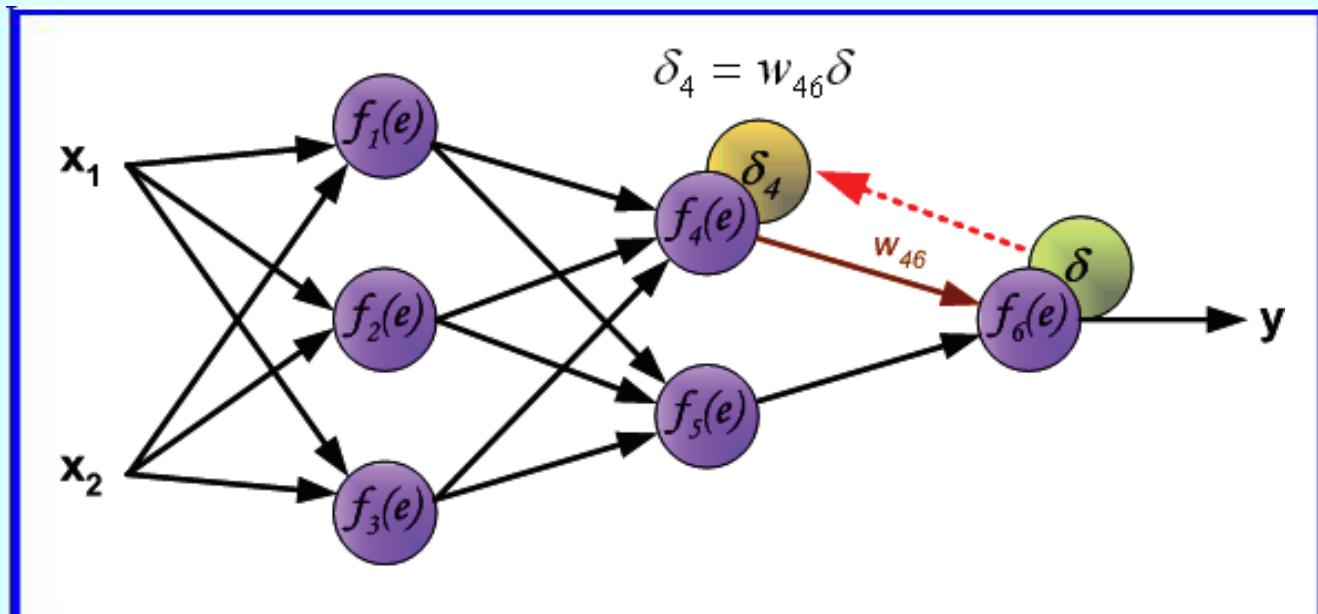


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 8)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (IX)

Пример работы алгоритма обратного распространения ошибки – 9

Обратное распространение ошибки в сети (1)

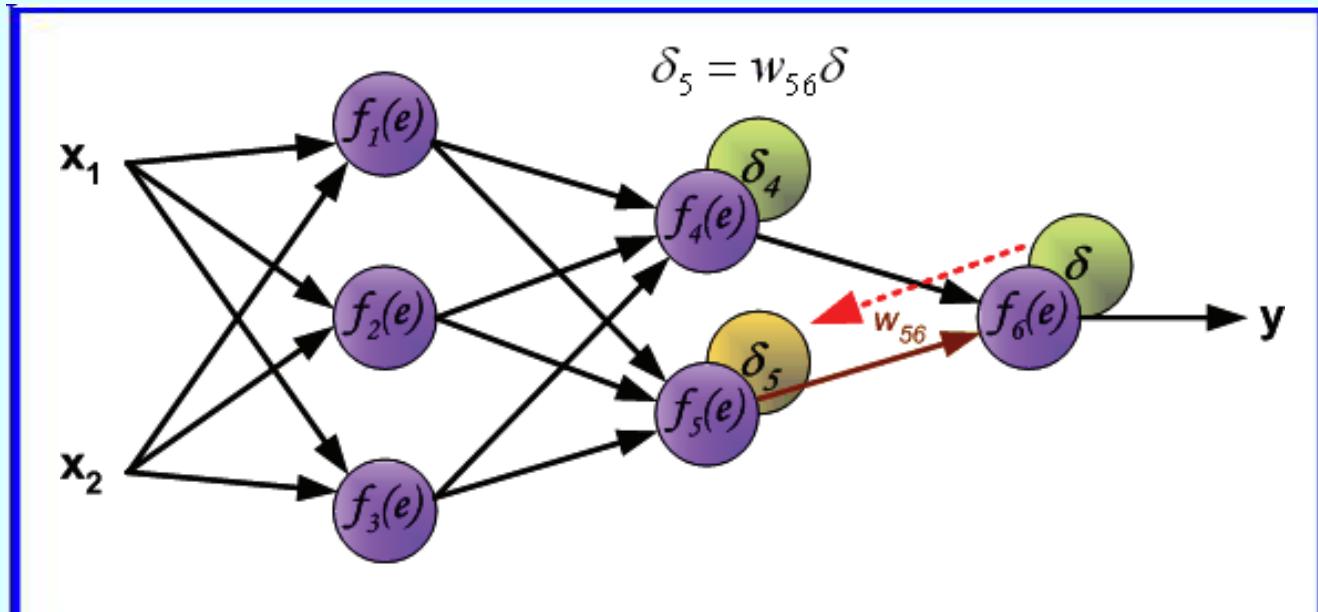


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 9)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (X)

Пример работы алгоритма обратного распространения ошибки – 10

Обратное распространение ошибки в сети (2)

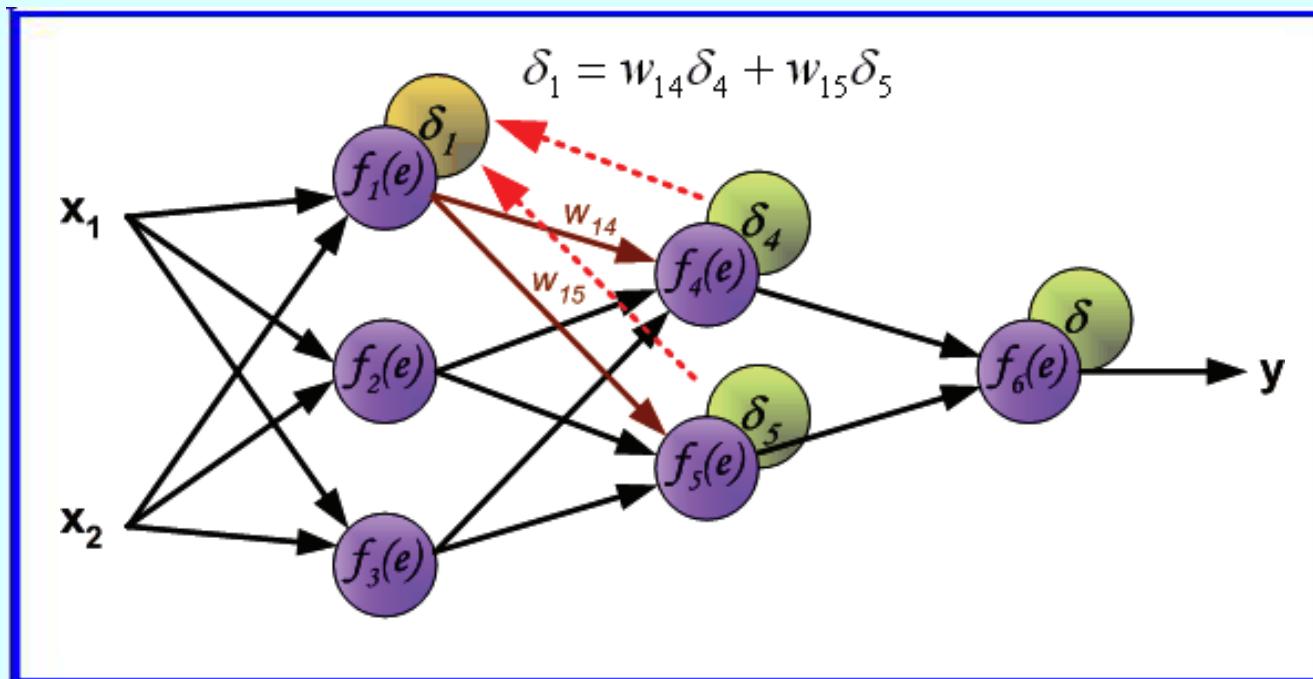


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 10)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XI)

Пример работы алгоритма обратного распространения ошибки – 11

Обратное распространение ошибки в сети (3)

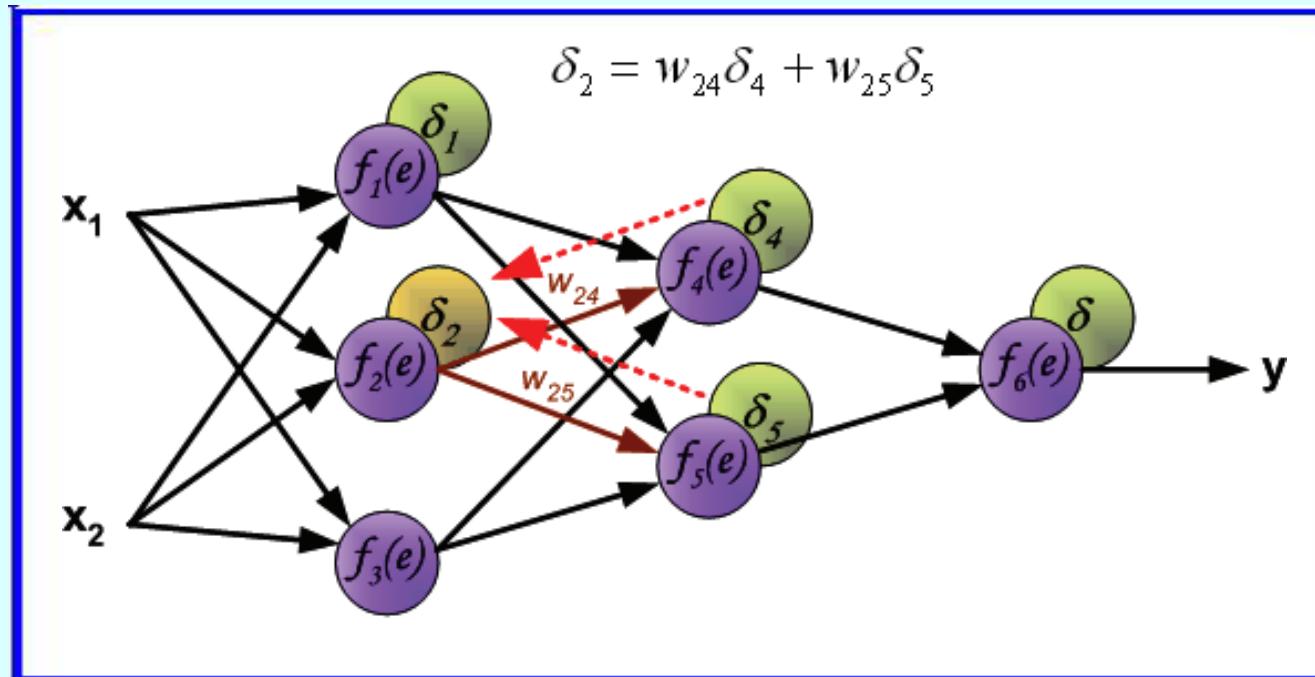


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 11)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XII)

Пример работы алгоритма обратного распространения ошибки – 12

Обратное распространение ошибки в сети (4)

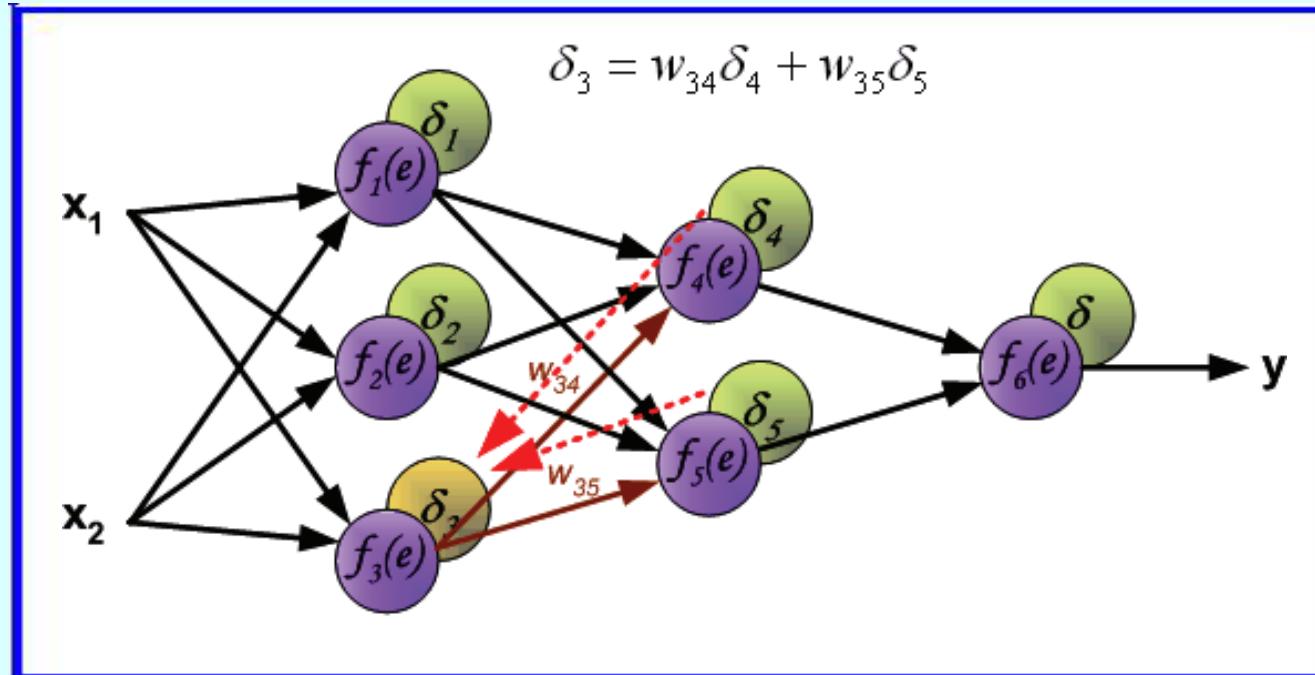


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 12)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XIII)

Пример работы алгоритма обратного распространения ошибки – 13

Обратное распространение ошибки в сети (5)

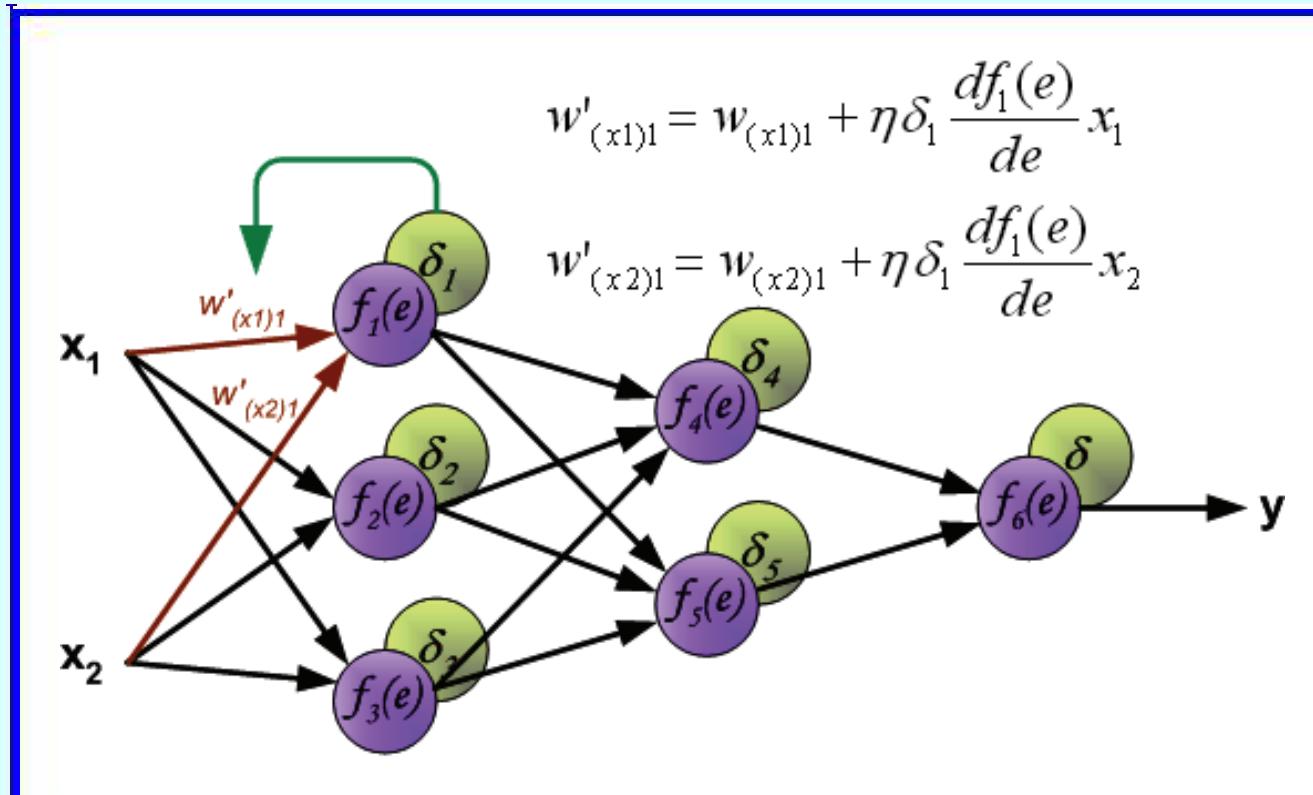


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 13)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XIV)

Пример работы алгоритма обратного распространения ошибки – 14

Корректировка весов связей (1)

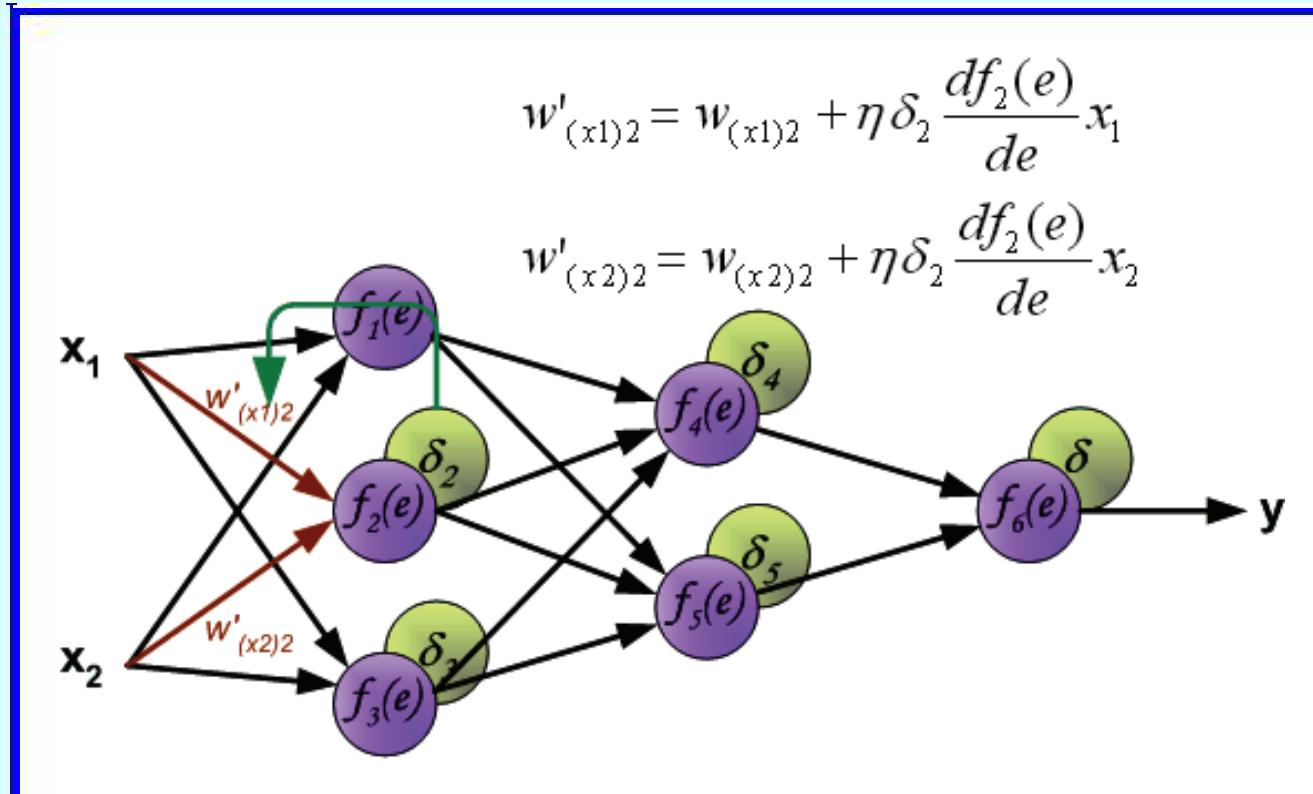


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 14)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XV)

Пример работы алгоритма обратного распространения ошибки – 15

Корректировка весов связей (2)

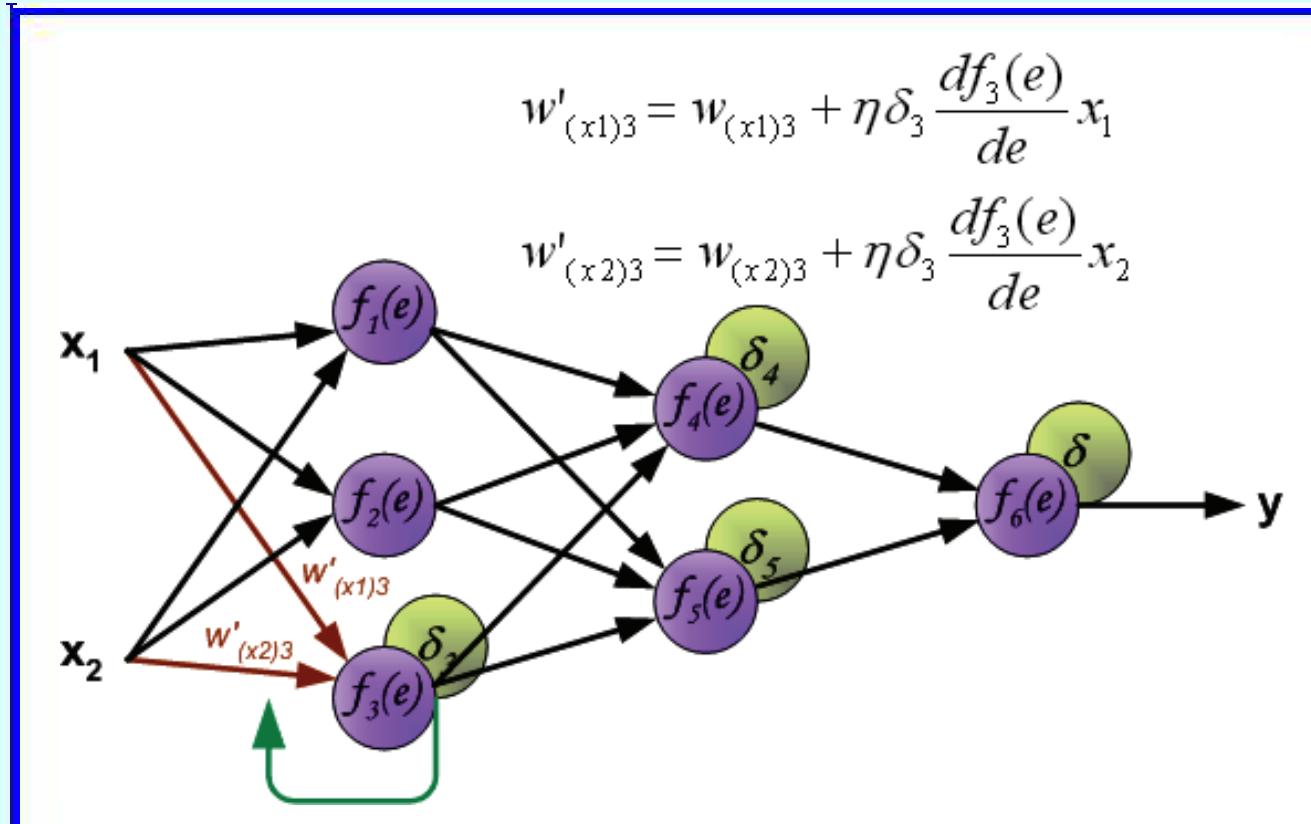


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 15)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XVI)

Пример работы алгоритма обратного распространения ошибки – 16

Корректировка весов связей (3)

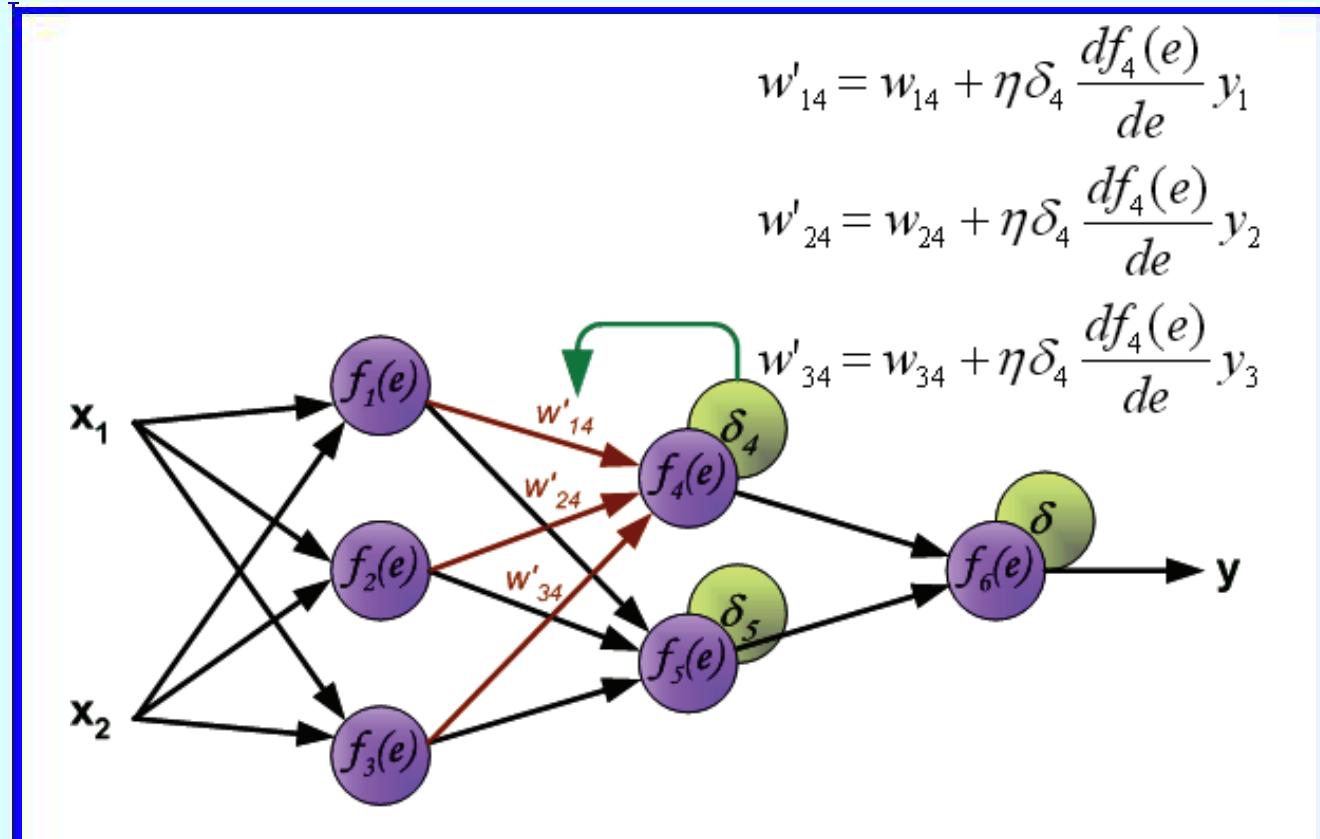


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 16)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XVII)

Пример работы алгоритма обратного распространения ошибки – 17

Корректировка весов связей (4)

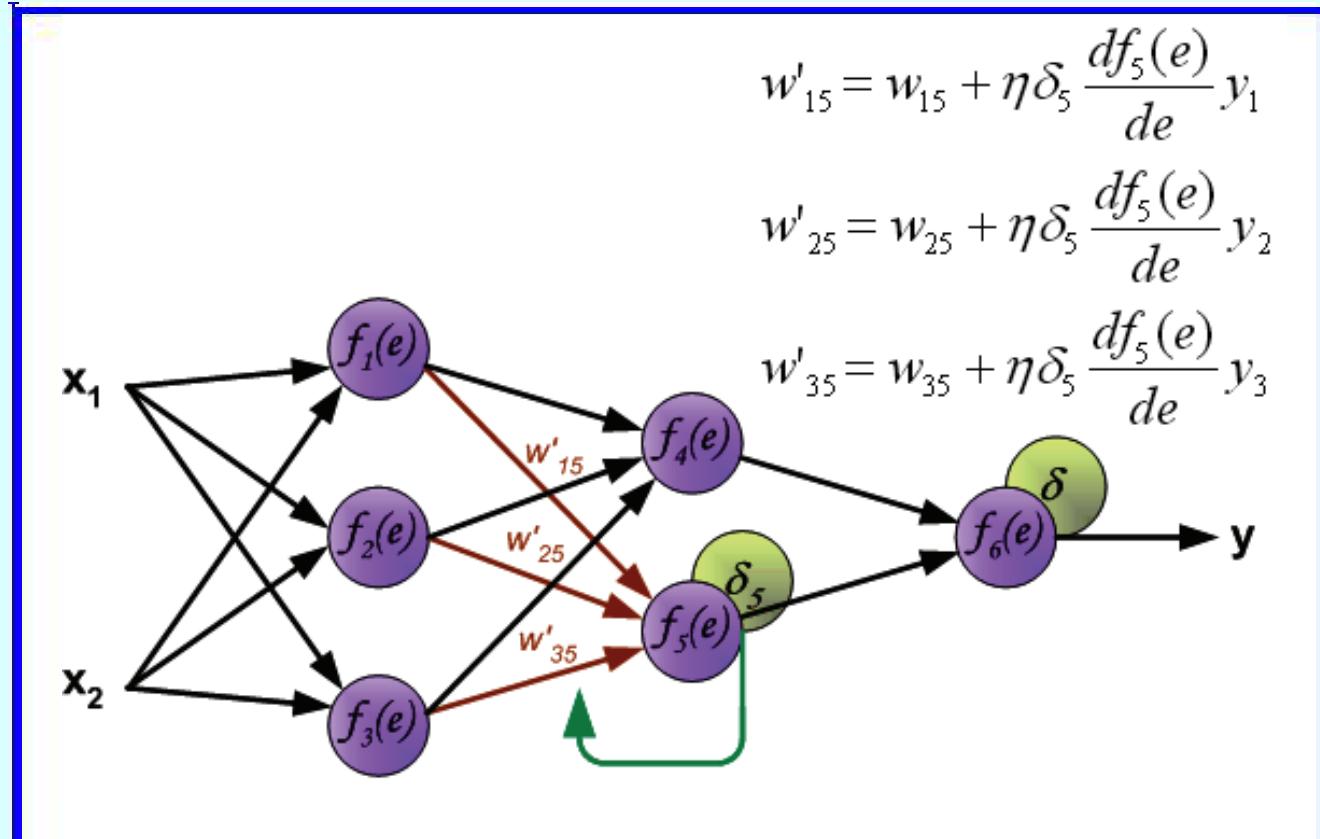


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 17)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XVIII)

Пример работы алгоритма обратного распространения ошибки – 18

Корректировка весов связей (5)

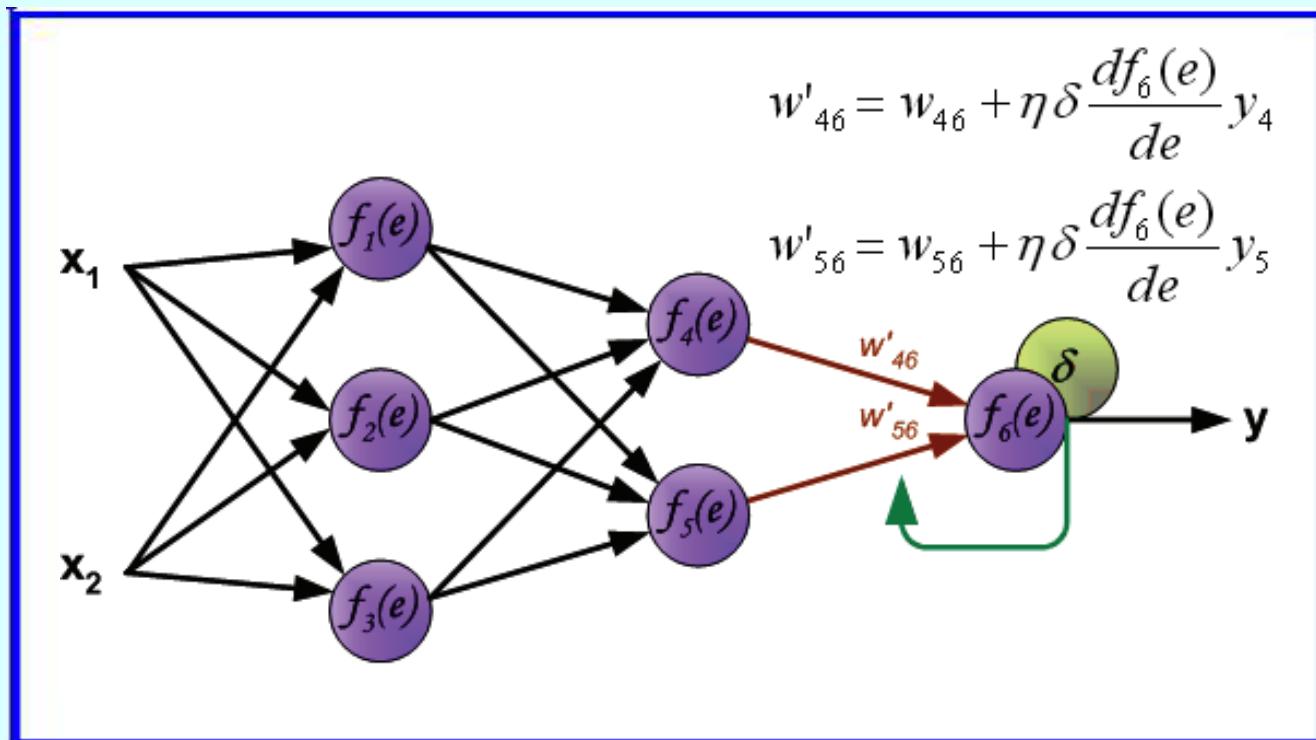


Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 18)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (XIX)

Пример работы алгоритма обратного распространения ошибки – 19

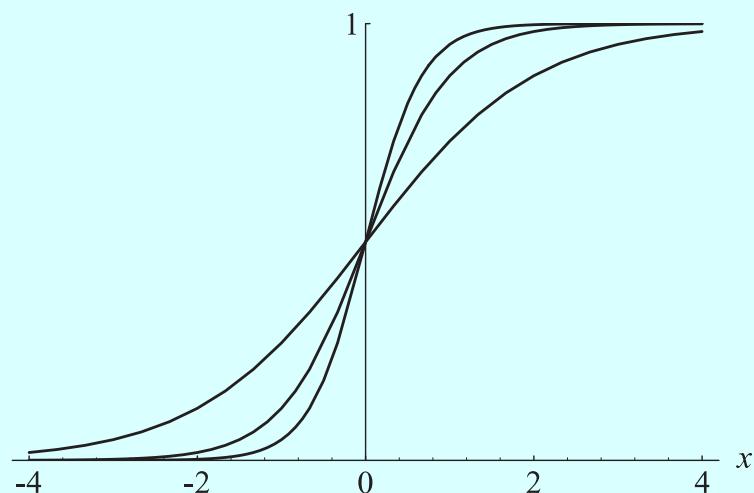
Корректировка весов связей (6)



Источник: *Bemacki M., Włodarczyk P.* Principles of training
for multi-layer neural network using backpropagation (Fig 19)
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Слоистые сети прямого распространения (ХХ)

Графическая форма алгоритма обратного распространения – 1



$$c = 1, \quad c = 2, \quad c = 3$$

**Сигмоидальная активационная
функция**
(логистическая функция)

$$s_c(x) = \frac{1}{1 + e^{-cx}}$$

С ростом c крутизна сигмоиды возрастает.

$$s_1(x) = s(x) = \frac{1}{1 + e^{-cx}}$$

Производная логистической функции:

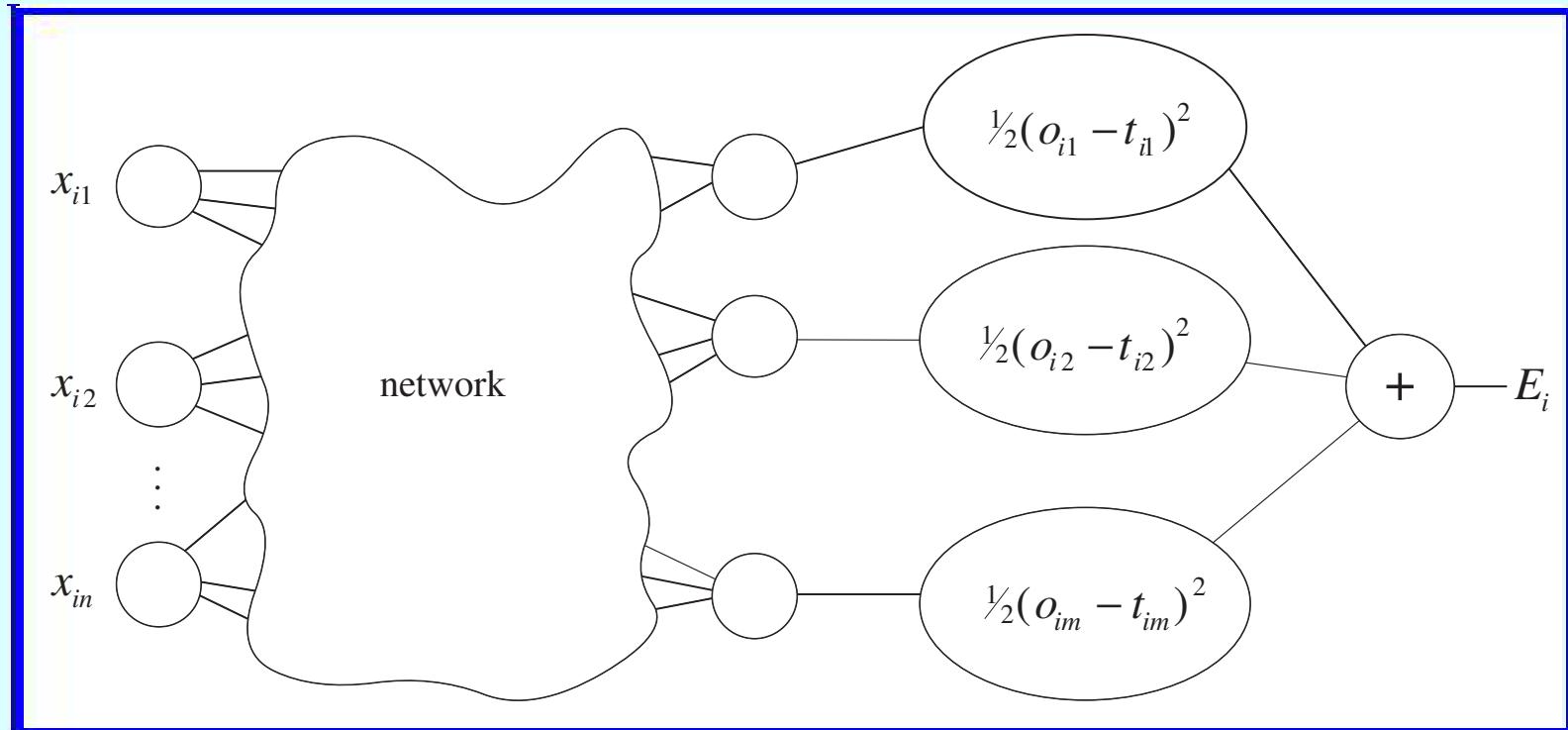
$$\frac{d}{dx} s(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = s(x)(1 - s(x))$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.1, p. 152)

Слоистые сети прямого распространения (ХI)

Графическая форма алгоритма обратного распространения – 2

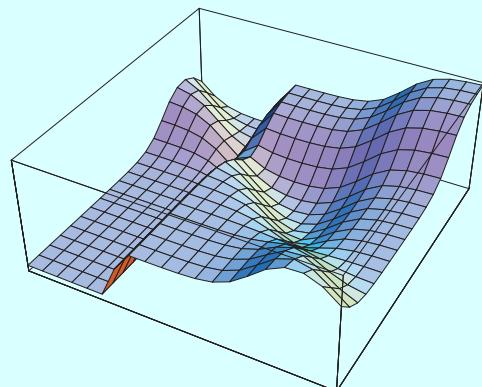
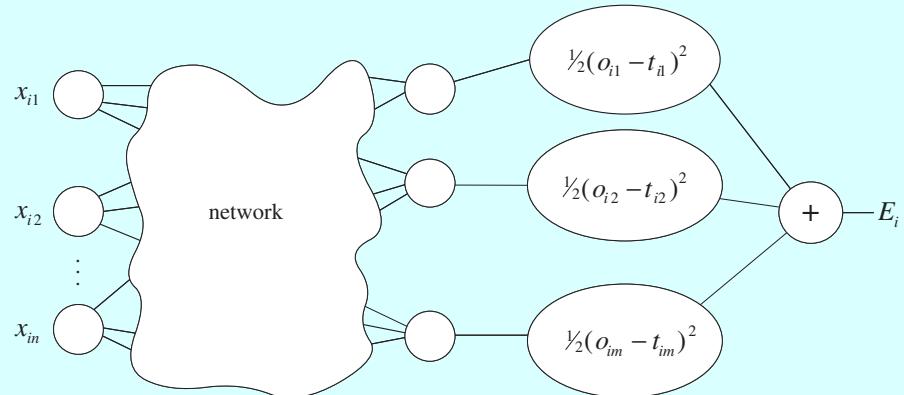
Расширенная сеть для вычисления функции ошибки



Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.6, p. 156)

Слоистые сети прямого распространения (XXII)

Графическая форма алгоритма обратного распространения – 3



$E(\mathbf{W})$: 1 нейрон, 2 входа, 4 примера

Обучающий набор:

$$\{(x_1, t_1), \dots, (x_p, t_p)\}$$

Функция ошибки сети:

$$E(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^p \|\mathbf{o}_i(\mathbf{W}) - \mathbf{t}_i\|^2$$

\mathbf{o}_i — вектор выходов сети

Корректировка весов сети:

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_L} \right)$$

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i}$$

γ — скорость обучения сети

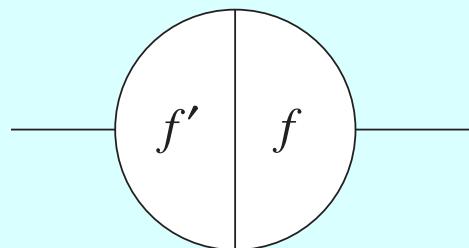
Проблема:

Вычисление градиента ∇E

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.6, p. 156; Figure 7.5, p. 155)

Слоистые сети прямого распространения (ХХIII)

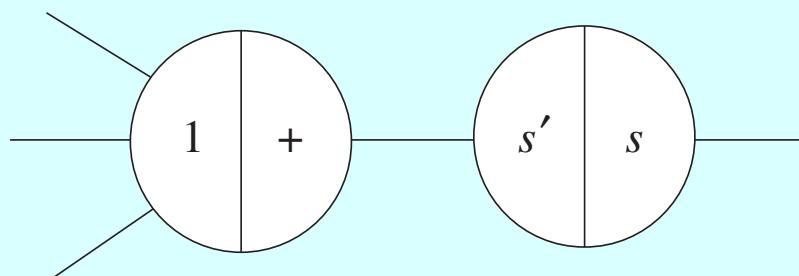
Графическая форма алгоритма обратного распространения – 4



Задача:

Построить эффективный метод вычисления **градиента** функции, заданной в сетевой форме.

f, f' – функция-примитив,
реализуемая нейроном
и ее производная



Структура функции-примитива:
Композиция сумматора
и активационной функции.

s, s' – активационная функция
нейрона и ее производная

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.7, 7.8, p. 158)

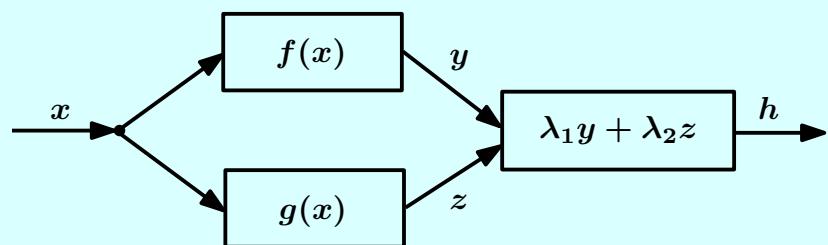
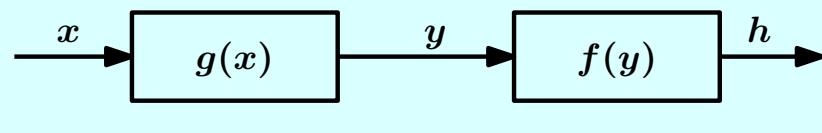
Слоистые сети прямого распространения (XXIV)

Графическая форма алгоритма обратного распространения – 5

Структурные модули сетевой модели

«Строительные блоки» трех видов:

- композиция функций;
- суммирование функций;
- межнейронная связь.



Композиция функций:

$$y = g(x), \quad h = f(y)$$

$$h(x) = (g \circ f)(x) = f(g(x))$$

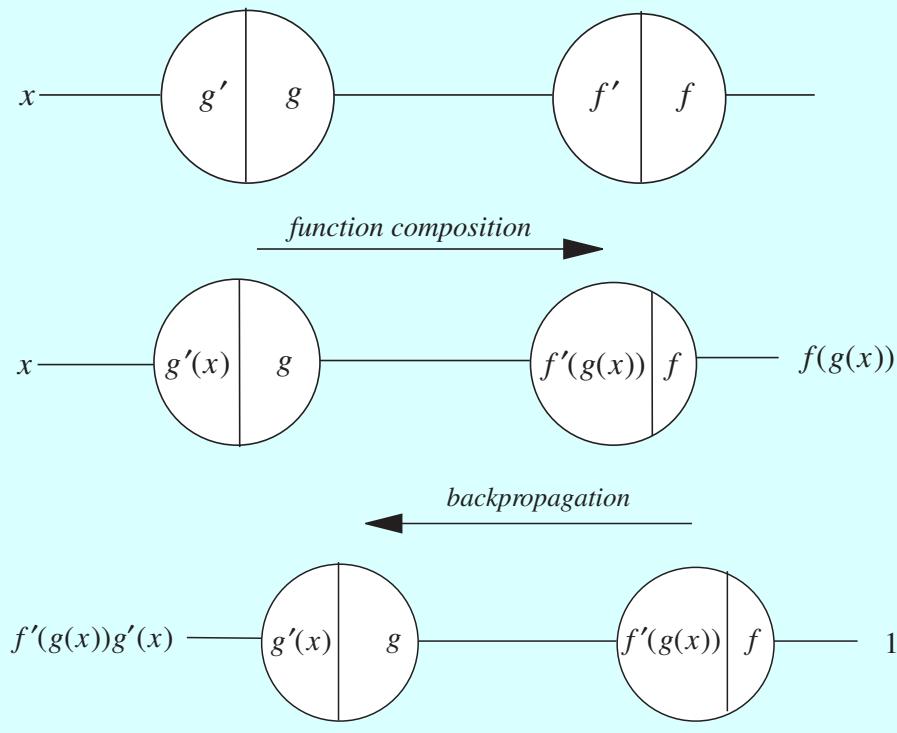
Суммирование функций:

$$y = f(x), \quad z = g(x)$$

$$h(x) = \lambda_1 f(x) + \lambda_2 g(x)$$

Слоистые сети прямого распространения (ХХV)

Графическая форма алгоритма обратного распространения – 6



**Композиция двух функций
в сетевом виде**

$$f \circ g = f(g(x))$$

**Прямое распространение
входного сигнала в сети**
(работают правые части узлов сети)

$$x \Rightarrow g(x) \Rightarrow f(g(x))$$

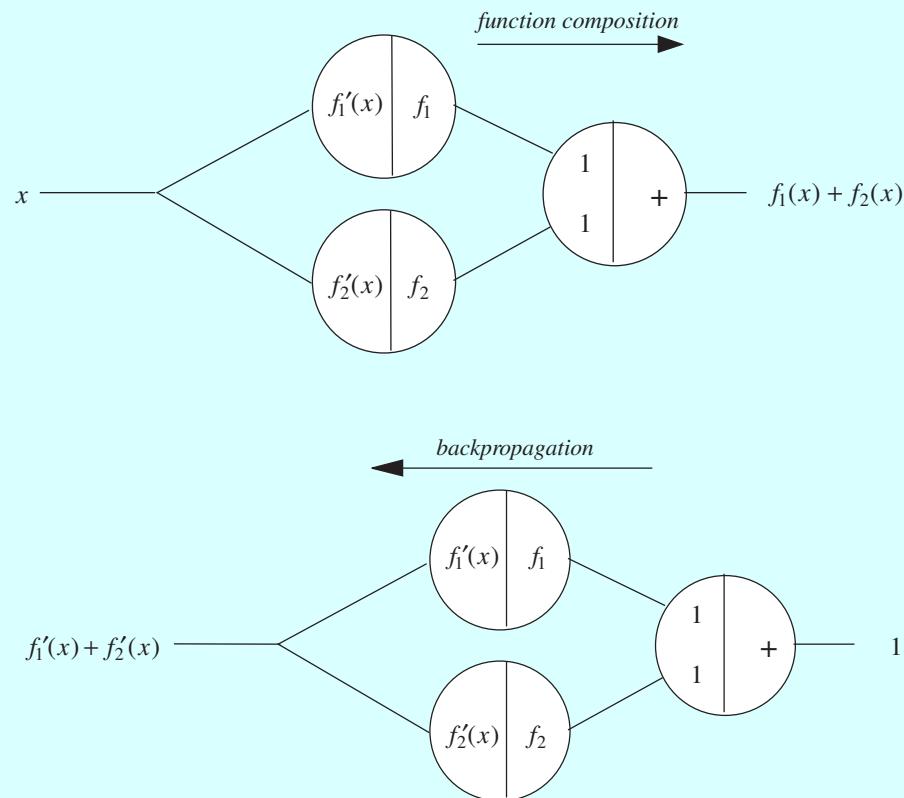
**Обратное распространение
выходного сигнала в сети**
(работают левые части узлов сети,
реализуется **цепное правило**
дифференцирования)

$$1 \Rightarrow f'(g(x)) \Rightarrow f'(g(x))g'(x)$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.9, 7.10, 7.11, p. 159)

Слоистые сети прямого распространения (XXVI)

Графическая форма алгоритма обратного распространения – 7



**Сложение двух функций
в сетевом виде**

$$f_1(x) + f_2(x)$$

**Прямое распространение
входного сигнала в сети**

$$\begin{aligned} x \Rightarrow & \{f_1(x), f_2(x)\} \Rightarrow \\ & \Rightarrow f_1(x) + f_2(x) \end{aligned}$$

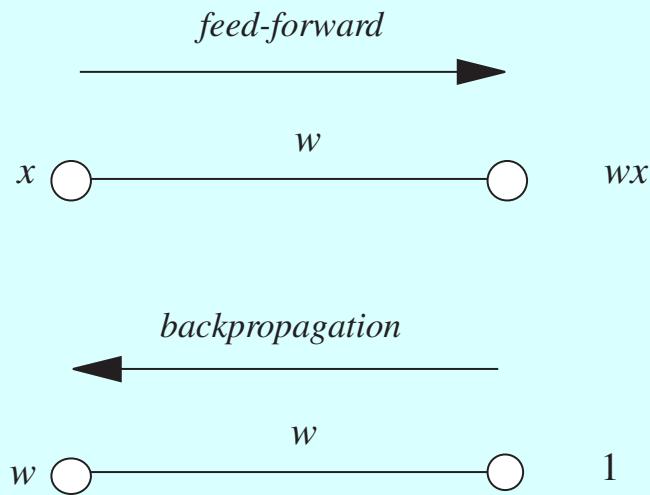
**Обратное распространение
выходного сигнала в сети**

$$\begin{aligned} 1 \Rightarrow & \{f'_1(x), f'_2(x)\} \Rightarrow \\ & \Rightarrow f'_1(x) + f'_2(x) \end{aligned}$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figures 7.12, 7.13, pp. 159, 160)

Слоистые сети прямого распространения (XXVII)

Графическая форма алгоритма обратного распространения – 8



**Взвешенные связи
в сети**

**Прямое распространение
входного сигнала в сети**

$$x \Rightarrow x \cdot w \Rightarrow wx$$

**Обратное распространение
выходного сигнала в сети**

$$1 \Rightarrow 1 \cdot w \Rightarrow w$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.14, p. 160)

Слоистые сети прямого распространения (XXVIII)

Графическая форма алгоритма обратного распространения – 9

Общая схема алгоритма в графической интерпретации:

Прямое распространение: На вход сети подается величина x , затем:

- вычисляются функции-примитивы и их производные в узлах (элементах) сети;
- значения производных запоминаются для последующего использования на этапе обратного распространения.

Обратное распространение: На выход сети подается сигнал 1, пропускаемый в направлении ко входу сети, затем:

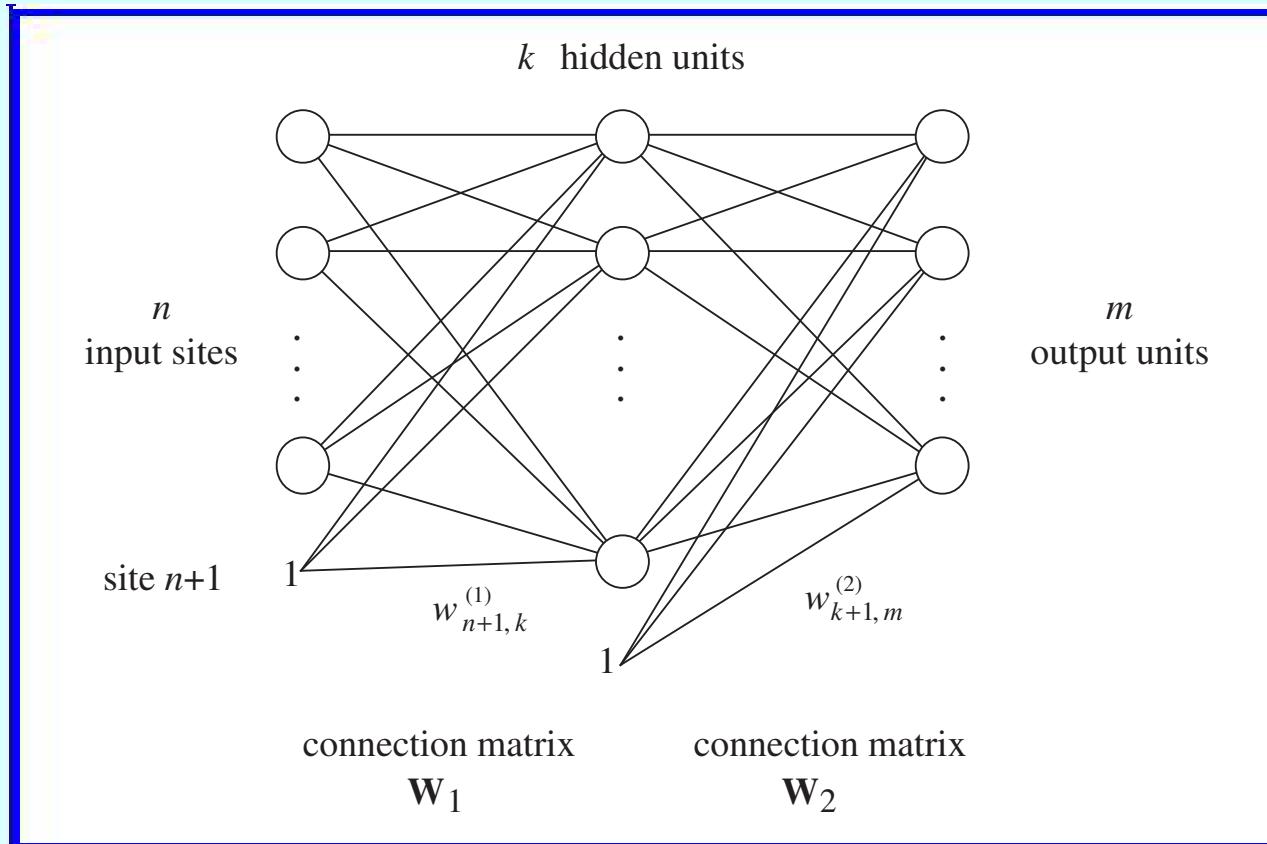
- информация, поступающая в элемент сети, суммируется, далее полученный результат умножается на величину, сохраненную в левой части узла сети, после чего передается далее в направлении ко входу сети;
- результат, накопленный во входном узле сети, представляет собой производную dF/dx для сетевой функции $F(x)$.

Здесь было принято, что сеть реализует функцию $y = F(x)$
(у сети один скалярный вход и один скалярный выход).

Слоистые сети прямого распространения (XXIX)

Графическая форма алгоритма обратного распространения – 10

Слоистые сети (1)

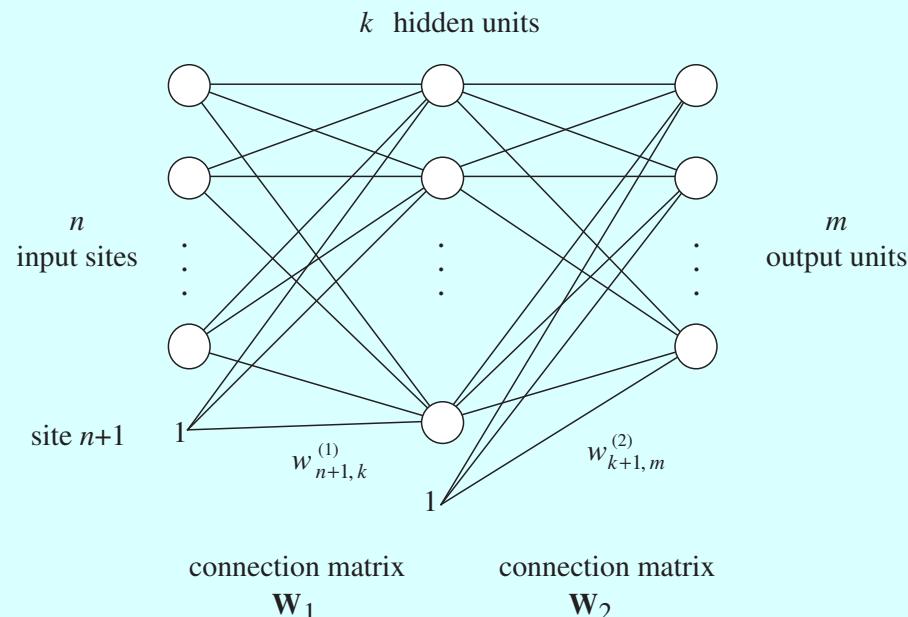


Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.17, p. 165)

Слоистые сети прямого распространения (XXX)

Графическая форма алгоритма обратного распространения – 11

Слоистые сети (2)



Параметры сети:

- n — число элементов во **входном** слое
- k — число элементов в **скрытом** слое
- m — число элементов в **выходном** слое

Матрица размерности $(n + 1) \times k$ весов связей между входным и скрытым слоями:

$$\widetilde{W}_1 = ||w_{ij}^{(1)}||$$

Матрица размерности $(k + 1) \times m$ весов связей между скрытым и входным слоями:

$$\widetilde{W}_2 = ||w_{ij}^{(2)}||$$

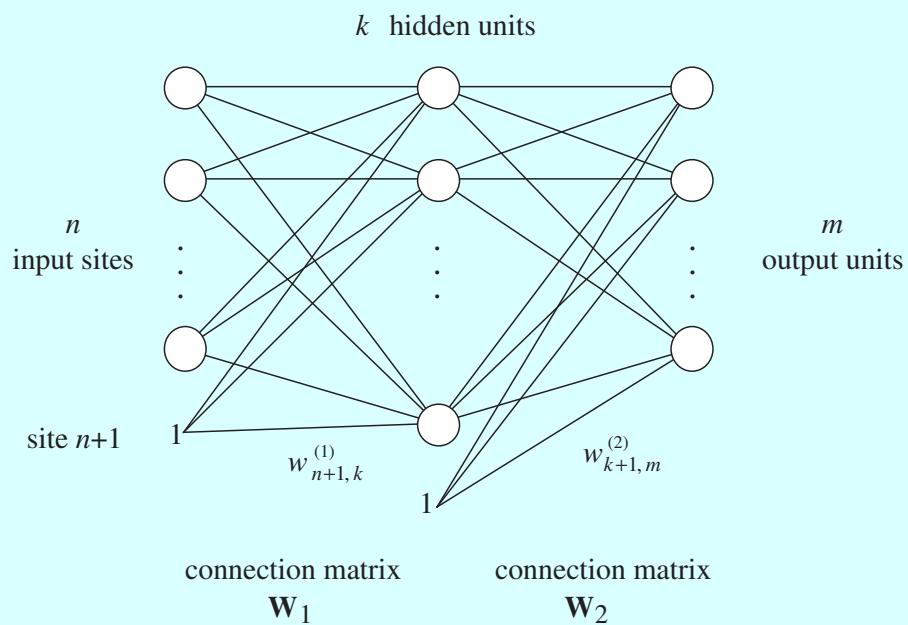
Матрицы \widetilde{W}_1 и \widetilde{W}_2 в качестве последней строки включают в себя **вектор смещений** для скрытого и выходного слоя, соответственно.

Матрицы W_1 и W_2 — **без** такой строки.

Слоистые сети прямого распространения (XXXI)

Графическая форма алгоритма обратного распространения – 12

Слоистые сети (3)



Расширенный входной вектор:
 $\hat{o}_i = (o_1, \dots, o_n, 1)$

Выход сумматора j -го элемента скрытого слоя:

$$net_j = \sum_{i=1}^{n+1} w_{ij}^{(1)} \hat{o}_i$$

Выход j -го элемента скрытого слоя:

$$\sigma_j^{(1)} = s\left(\sum_{i=1}^{n+1} w_{ij}^{(1)} \hat{o}_i\right)$$

s — сигмоида

Векторный выход скрытого слоя:

$$\mathbf{o}^{(1)} = s(\hat{\mathbf{o}} \widetilde{\mathbf{W}}_1)$$

Векторный выход сети:

$$\mathbf{o}^{(2)} = s(\hat{\mathbf{o}}^{(1)} \widetilde{\mathbf{W}}_2)$$

Слоистые сети прямого распространения (XXXII)

Графическая форма алгоритма обратного распространения – 13

Перед началом работы алгоритма осуществляется **инициализация** весов сети небольшими случайными значениями.

Шаги алгоритма:

(выполняются циклически до выполнения условия завершения)

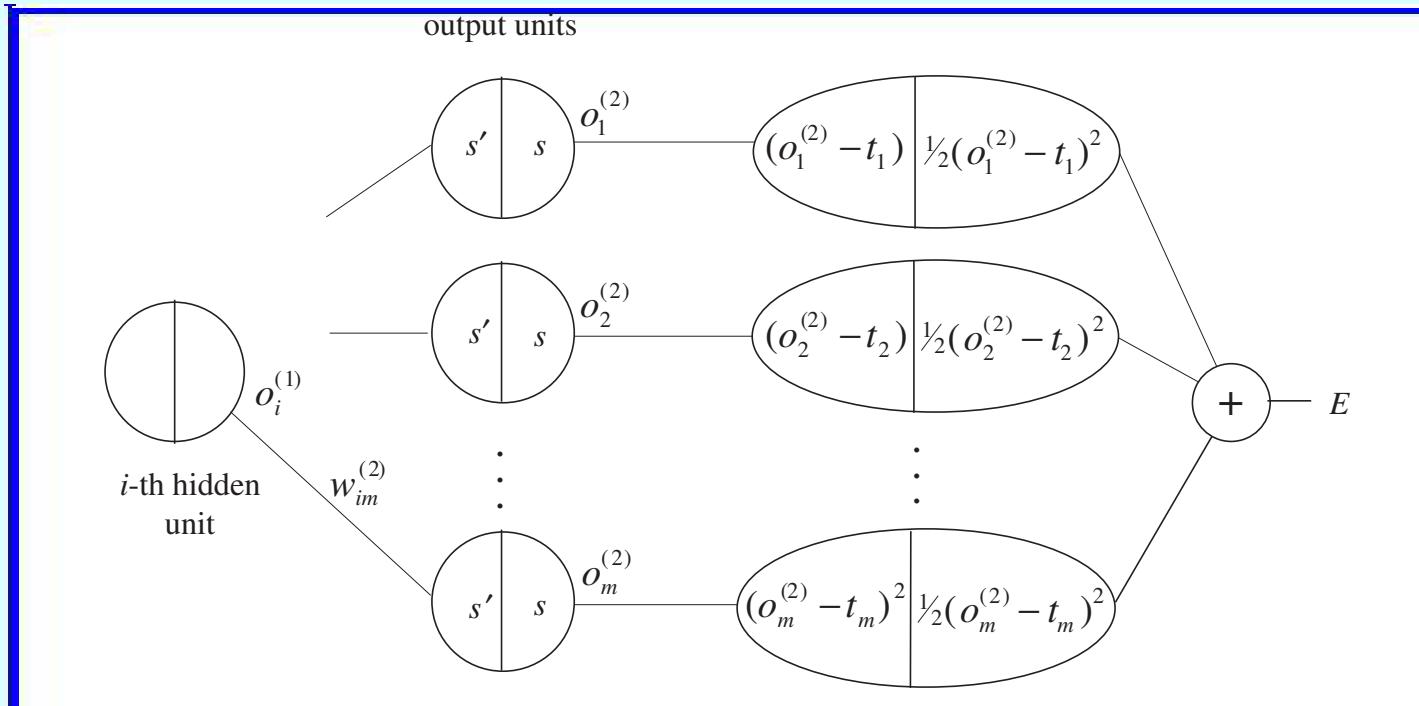
- 1) Вычисления в **прямом направлении** (от входов к выходам сети)
- 2) Обратное распространение ошибки (невязки) к **выходному слою**
- 3) Обратное распространение ошибки (невязки) к **скрытому слою**
- 4) **Корректировка** весов сети

Условие завершения алгоритма — ошибка сети должна стать достаточно малой (не превышать некоторого наперед заданного значения).

Слоистые сети прямого распространения (XXXIII)

Графическая форма алгоритма обратного распространения – 14

Шаг 1: Вычисления в прямом направлении



На вход сети подается вектор \mathbf{o} .

Вычисляются и запоминаются векторы $\mathbf{o}^{(1)}$ и $\mathbf{o}^{(2)}$ — выходы скрытого слоя и сети, соответственно.

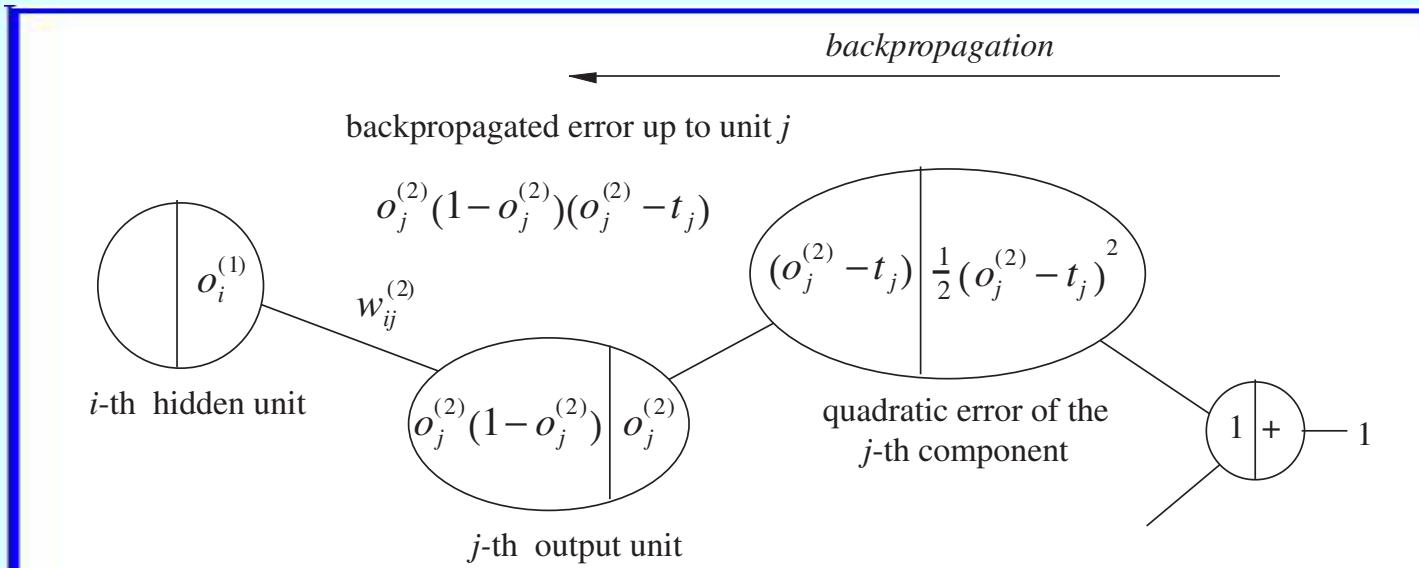
Для каждого элемента (нейрона) сети вычисляется и запоминается производная s' активационной функции.

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.18, p. 167)

Слоистые сети прямого распространения (XXXIV)

Графическая форма алгоритма обратного распространения – 15

Шаг 2: Обратное распространение ошибки к выходному слою



Вычисление частной производной $\partial E / \partial w_{ij}^{(2)}$:

Невязка для j -го элемента выходного слоя: $\delta_j^{(2)} = o_j^{(2)}(1 - o_j^{(2)})(o_j^{(2)} - t_j)$

Искомая частная производная:

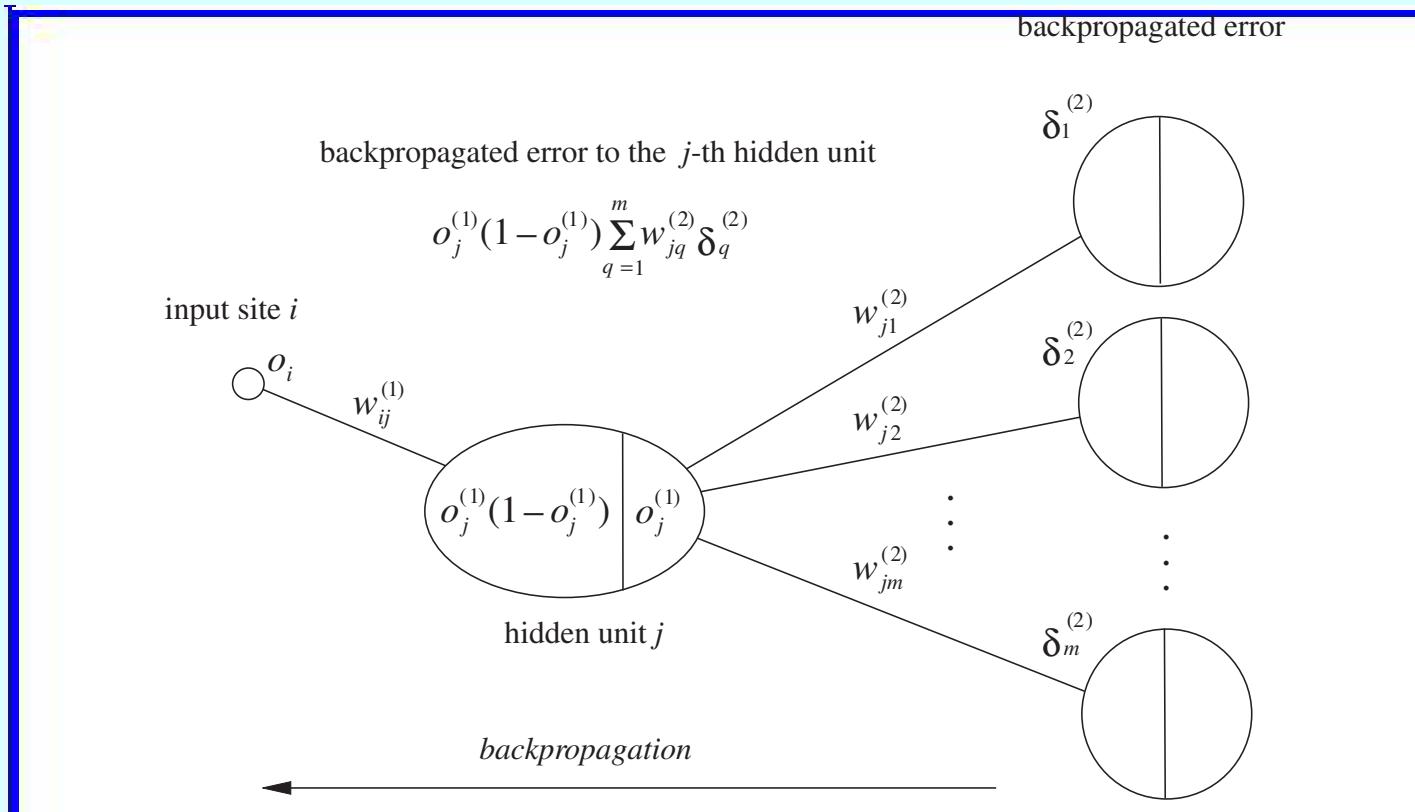
$$\partial E / \partial w_{ij}^{(2)} = [o_j^{(2)}(1 - o_j^{(2)})(o_j^{(2)} - t_j)]o_i^{(1)} = \delta_j^{(2)}o_i^{(1)}$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.19, p. 167)

Слоистые сети прямого распространения (XXXV)

Графическая форма алгоритма обратного распространения – 16

Шаг 3: Обратное распространение ошибки к скрытому слою (1)



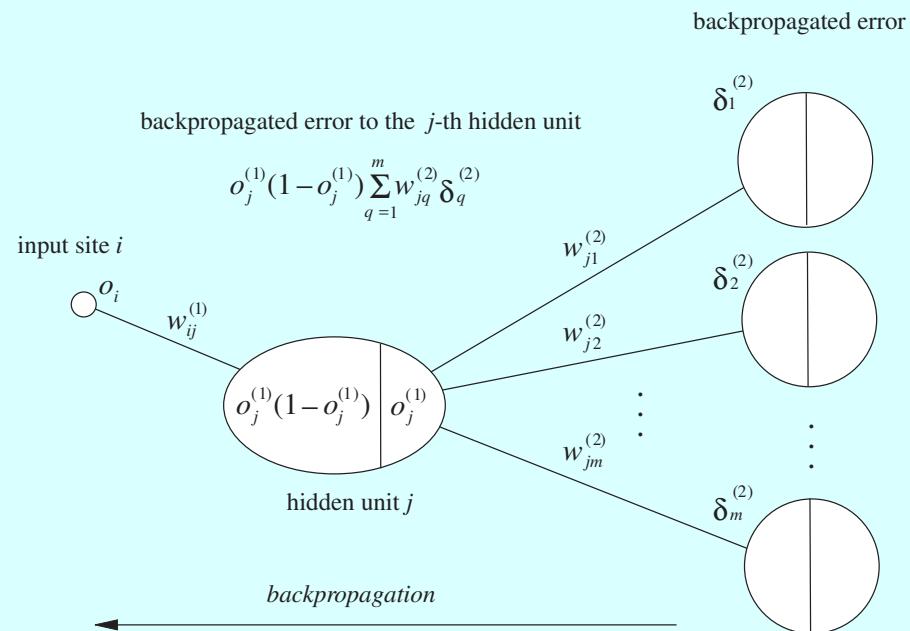
Вычисление частной производной $\partial E / \partial w_{ij}^{(1)}$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.21, p. 169)

Слоистые сети прямого распространения (XXXVI)

Графическая форма алгоритма обратного распространения – 17

Шаг 3: Обратное распространение ошибки к скрытому слою (2)



Вычисление частной производной
 $\partial E / \partial w_{ij}^{(1)}$

Невязка для j -го элемента
выходного слоя:

$$\delta_j^{(1)} = o_j^{(1)} (1 - o_j^{(1)}) \sum_{q=1}^m w_{jq}^{(2)} \delta_q^{(2)}$$

Искомая частная производная:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_j^{(1)} o_i$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 7.19, p. 167)

Слоистые сети прямого распространения (XXXVII)

Графическая форма алгоритма обратного распространения – 18

Шаг 4: Корректировка весов сети

Веса сети корректируются в направлении убывания функции ошибки $E(W)$, определяемом антиградиентом этой функции.

Корректировка весов связей между входным и скрытым слоями:

$$\Delta w_{ij}^{(1)} = -\gamma o_i \delta_j^{(1)}, \quad i = 1, \dots, n+1; \quad j = 1, \dots, k$$

Корректировка весов связей между скрытым и выходным слоями:

$$\Delta w_{ij}^{(2)} = -\gamma o_i^{(1)} \delta_j^{(2)}, \quad i = 1, \dots, k+1; \quad j = 1, \dots, m$$

Здесь γ — коэффициент скорости обучения

Принято также, что $o_{n+1} = o_{k+1}^{(1)} = 1$

Корректировка весов должна осуществлять **только после вычисления** ошибок **для всех** элементов сети.

Слоистые сети прямого распространения (XXXVIII)

Графическая форма алгоритма обратного распространения – 19

Матричная форма алгоритма (1)

Параметры сети:

n — число элементов во **входном** слое

k — число элементов в **скрытом** слое

m — число элементов в **выходном** слое

Обработка входного вектора сетью:

$$\mathbf{o} \Rightarrow [\mathbf{o}^{(1)} = s(\hat{\mathbf{o}}\tilde{\mathbf{W}}_1)] \Rightarrow [\mathbf{o}^{(2)} = s(\hat{\mathbf{o}}^{(1)}\tilde{\mathbf{W}}_2)]$$

Матрицы $\tilde{\mathbf{W}}_1$ и $\tilde{\mathbf{W}}_2$ в качестве последней строки включают в себя **вектор смещений** для скрытого и выходного слоя, соответственно.

Матрицы \mathbf{W}_1 и \mathbf{W}_2 (для шагов обратного распространения ошибки) **не содержат** такой строки.

Слоистые сети прямого распространения (XXXIX)

Графическая форма алгоритма обратного распространения – 20

Матричная форма алгоритма (2)

Производные, запоминаемые на шаге прямого распространения

Матрица производных для k элементов скрытого слоя:

$$D_1 = \begin{bmatrix} o_1^{(1)}(1 - o_1^{(1)}) & 0 & \dots & 0 \\ 0 & o_2^{(1)}(1 - o_2^{(1)}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & o_k^{(1)}(1 - o_k^{(1)}) \end{bmatrix}$$

Матрица производных для m элементов выходного слоя:

$$D_2 = \begin{bmatrix} o_1^{(2)}(1 - o_1^{(2)}) & 0 & \dots & 0 \\ 0 & o_2^{(2)}(1 - o_2^{(2)}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & o_m^{(2)}(1 - o_m^{(2)}) \end{bmatrix}$$

Слоистые сети прямого распространения (XL)

Графическая форма алгоритма обратного распространения – 21

Матричная форма алгоритма (3)

Производные, запоминаемые на шаге прямого распространения
и ошибки элементов сети

Вектор производных квадратичных составляющих ошибки сети:

$$\mathbf{e} = \begin{bmatrix} (o_1^{(2)} - t_1) \\ (o_2^{(2)} - t_2) \\ \vdots \\ (o_m^{(2)} - t_m) \end{bmatrix}$$

m -мерный вектор невязок элементов выходного слоя сети:

$$\delta^{(2)} = D_2 \mathbf{e}$$

k -мерный вектор невязок элементов скрытого слоя сети:

$$\delta^{(1)} = D_1 W_2 \delta^{(2)}$$

Слоистые сети прямого распространения (XLI)

Графическая форма алгоритма обратного распространения – 22

Матричная форма алгоритма (4)

Корректировка весовых матриц \widetilde{W}_1 и \widetilde{W}_2 :

$$\Delta \widetilde{W}_2^T = -\gamma \delta^{(2)} \hat{o}^{(1)}$$

$$\Delta \widetilde{W}_1^T = -\gamma \delta^{(1)} \hat{o}$$

Обобщение для случая сети из L слоев

Вектор невязок для элементов выходного слоя:

$$\delta^{(L)} = D_L e$$

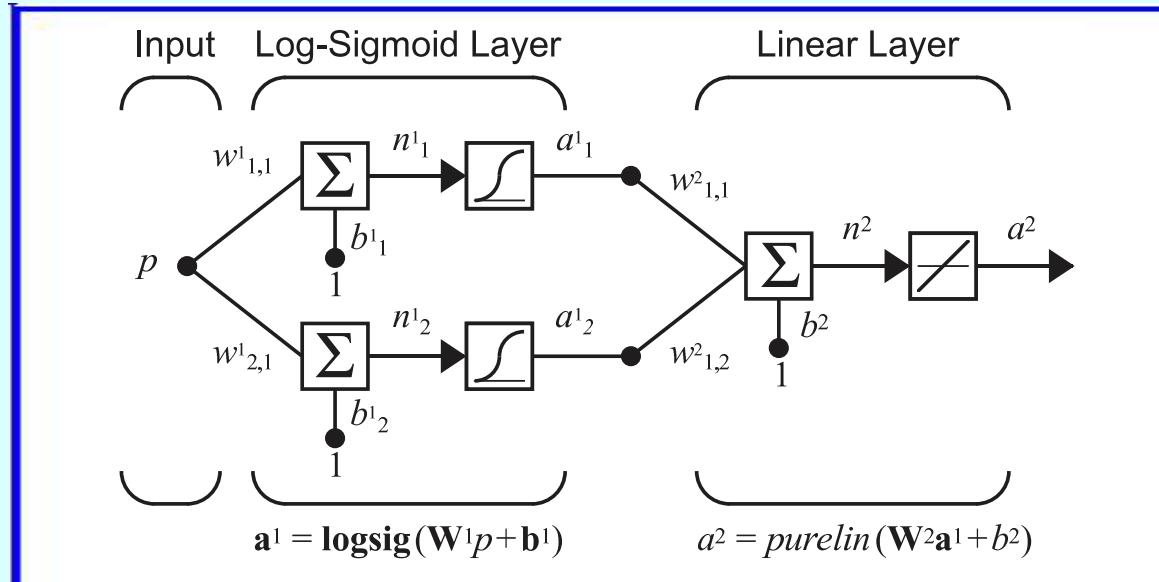
Вектор невязок для элементов i -го слоя:

$$\delta^{(i)} = D_i W_{i+1} \delta^{(i+1)}$$

W_{i+1} — матрица весов связей между i -м и $i + 1$ -м слоями,
 $i = 0, 1, \dots, L$; $i = 0$ — входной слой сети.

Применение сетей прямого распространения (I)

Апроксимация функции сетью 1-2-1 (1)



$$a_1^{(1)} = \text{logsig}(w_{1,1}^{(1)} p + b_1^{(1)})$$

$$a_2^{(1)} = \text{logsig}(w_{2,1}^{(1)} p + b_2^{(1)})$$

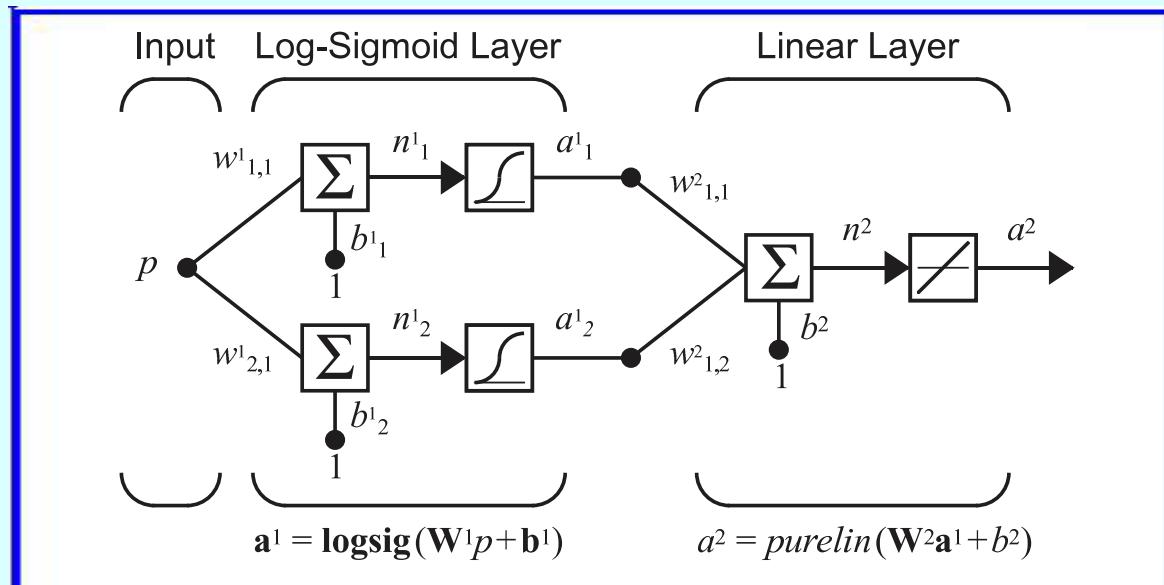
$$a^{(2)} = \text{purelin}(w_{1,1}^{(2)} a_1^{(1)} - w_{1,2}^{(1)} a_2^{(1)} + b^{(2)})$$

$$\text{logsig}(n) = \frac{1}{1 + e^{-n}}, \quad \text{purelin}(n) = n$$

Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.4, p.11-5).

Применение сетей прямого распространения (II)

Аппроксимация функции сетью 1-2-1 (2)



$$a_1^{(1)} = \text{logsig}(w_{1,1}^{(1)} p + b_1^{(1)}), \quad a_2^{(1)} = \text{logsig}(w_{2,1}^{(1)} p + b_2^{(1)})$$

$$a^{(2)} = \text{purelin}(w_{1,1}^{(2)} a_1^{(1)} - w_{1,2}^{(2)} a_2^{(1)} + b^{(2)})$$

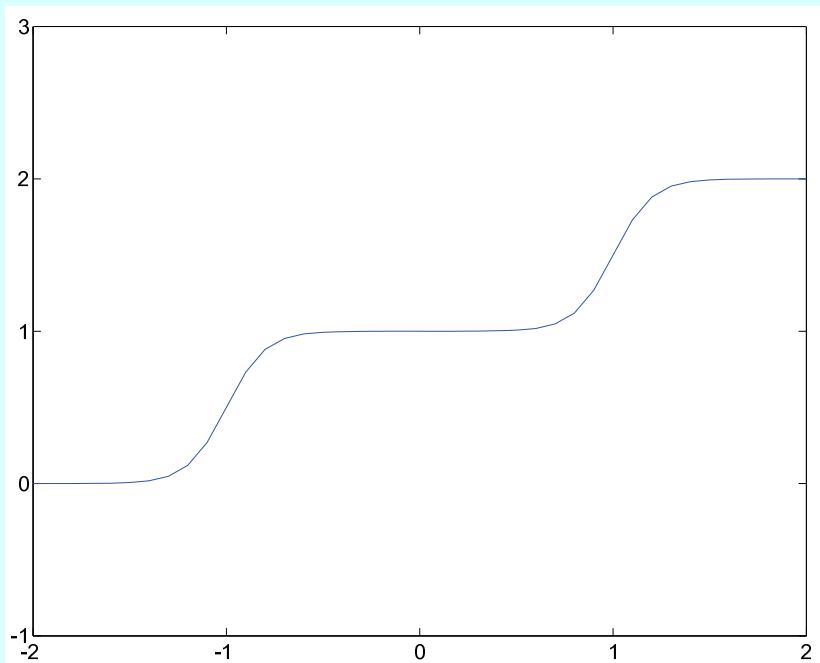
$$w_{1,1}^{(1)} = 10, \quad w_{2,1}^{(1)} = 10, \quad b_1^{(1)} = -10, \quad b_2^{(1)} = 10,$$

$$w_{1,1}^{(2)} = 1, \quad w_{1,2}^{(2)} = 1, \quad b^{(2)} = 0$$

$$p \in [-2, 2]$$

Применение сетей прямого распространения (III)

Аппроксимация функции сетью 1-2-1 (3)



Номинальный отклик сети

Центры «ступенек» сигмоид — при

$$n_{1,1}^{(1)} = n_{2,1}^{(1)} = 0$$

$$n_{1,1}^{(1)} = w_{1,1}^{(1)} p + b_1^{(1)} = 0$$

⇓

$$p = \frac{b_1^{(1)}}{w_{1,1}^{(1)}} = -\frac{-10}{10} = 1$$

$$n_{2,1}^{(1)} = w_{2,1}^{(1)} p + b_2^{(1)} = 0$$

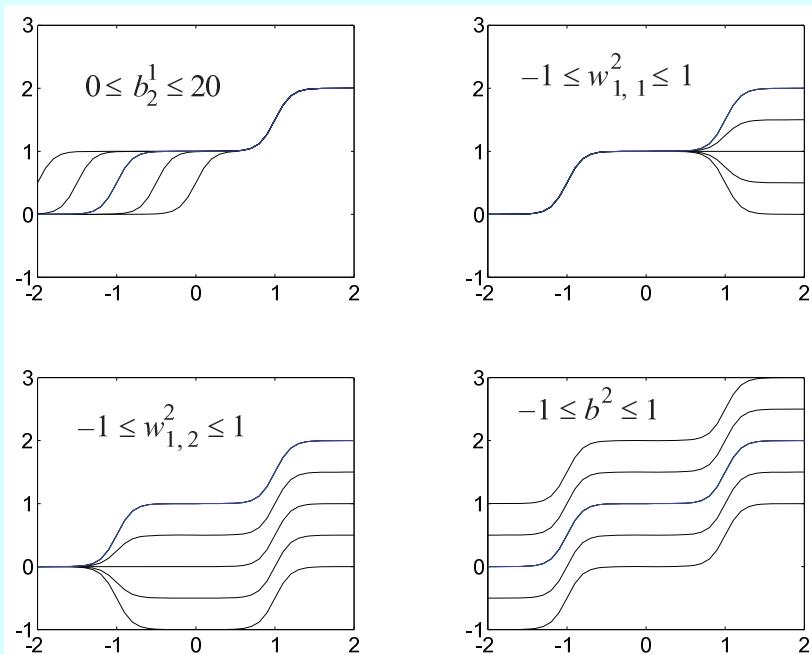
⇓

$$p = -\frac{b_2^{(1)}}{w_{2,1}^{(1)}} = -\frac{10}{10} = -1$$

Источник: *Hagan M. T., Demuth H.B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.5, p.11-6).

Применение сетей прямого распространения (IV)

Аппроксимация функции сетью 1-2-1 (4)



Гибкость многослойной сети
(синяя линия — номинальный отклик)
Диапазоны варьирования
некоторых параметров сети:

$$-1 \leq w_{1,1}^{(2)} \leq 1$$

$$-1 \leq w_{1,2}^{(2)} \leq 1$$

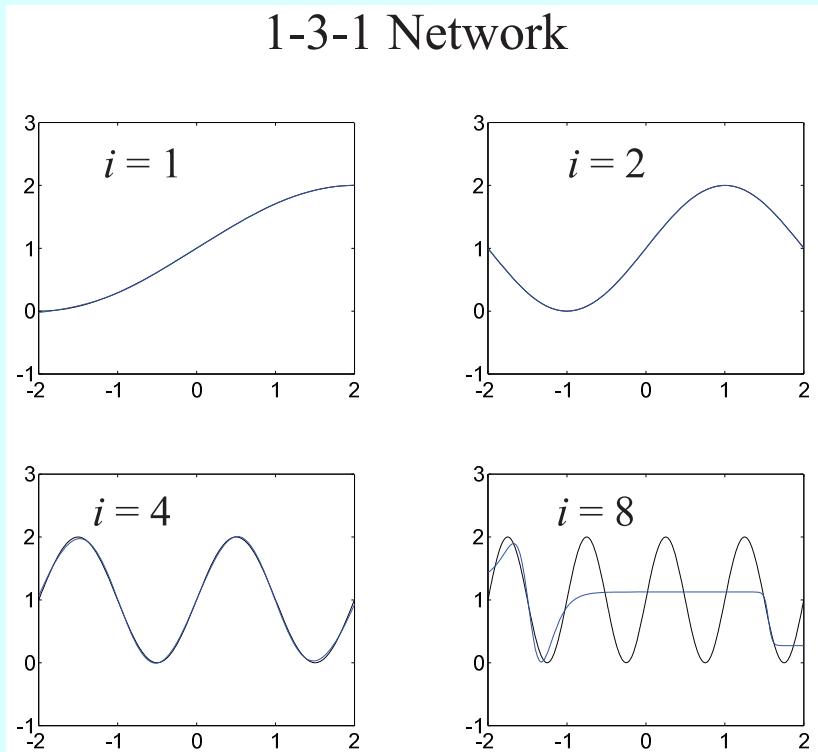
$$0 \leq b_2^{(1)} \leq 20$$

$$-1 \leq b^{(2)} \leq 1$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.6, p.11-7).

Применение сетей прямого распространения (V)

Выбор архитектуры сети – 1



Аппроксимация функции сетью 1-3-1

(синяя линия — отклик сети)

Аппроксимируемая функция:

$$g(p) = 1 + \sin\left(\frac{i\pi}{4}p\right),$$
$$-2 \leq p \leq 2$$

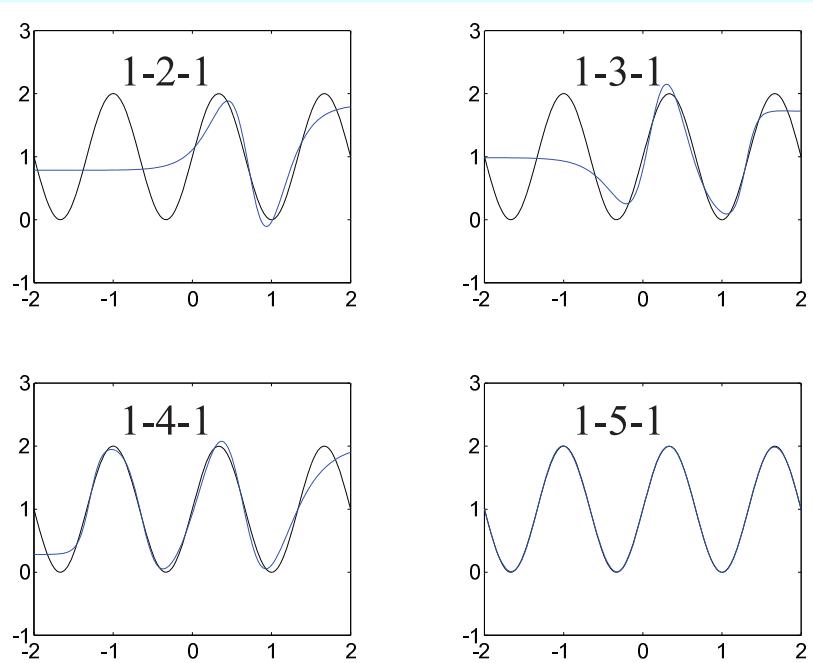
Вывод: При усложнении аппроксимируемой функции с определенного момента ($i = 8$) возможностей сети с выбранной архитектурой (1-3-1) становится недостаточно для решения задачи с требуемой точностью.

Вопрос: Как следует изменить сеть, чтобы она решила поставленную задачу?

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.10, p.11-18).

Применение сетей прямого распространения (VI)

Выбор архитектуры сети – 2



Влияние числа нейронов

в скрытом слое сети

(синяя линия — отклик сети)

Аппроксимируемая функция:

$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right),$$

$$-2 \leqslant p \leqslant 2$$

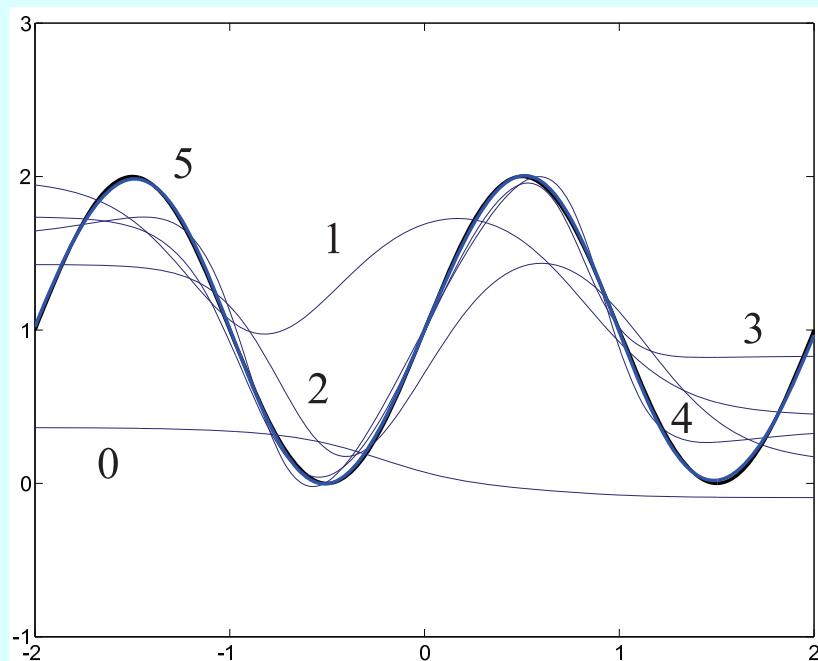
Вывод: В данном случае **возможности сети** становятся адекватными решаемой задаче, т.е. позволяют получить ее **решение с требуемой точностью**, при числе нейронов в скрытом слое, **равном 5**.

Вопрос: Целесообразно ли **дальнейшее наращивание** числа нейронов в скрытом слое сети?

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.11, p.11-19).

Применение сетей прямого распространения (VII)

Сходимость процесса обучения сети – 1



Сходимость к глобальному минимуму
функции ошибки сети
(сеть со структурой 1-3-1)

Апроксимируемая функция:
(утолщенная синяя линия)

$$g(p) = 1 + \sin(\pi p),
-2 \leq p \leq 2$$

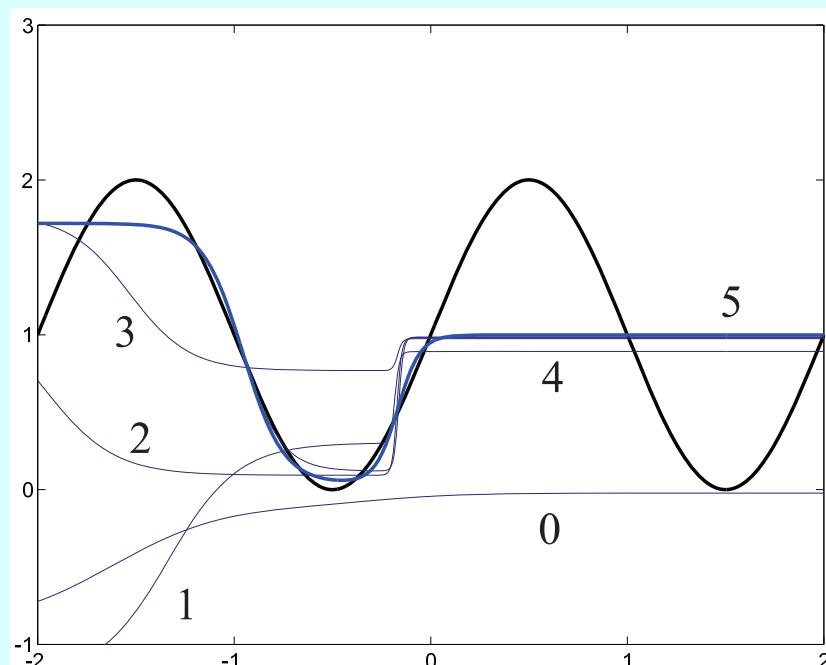
Пронумерованные кривые — **последовательность состояний** процесса обучения сети (числа рядом с кривыми — **НЕ** номера итераций процесса).

Завершающее состояние процесса (**с номером 5**) — утолщенная синяя линия.

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.12, p.11-20).

Применение сетей прямого распространения (VIII)

Сходимость процесса обучения сети – 2



Сходимость к локальному минимуму функции ошибки сети (сеть со структурой 1-3-1)

Апроксимируемая функция:
(утолщенная синяя линия)

$$g(p) = 1 + \sin(\pi p),$$

$$-2 \leqslant p \leqslant 2$$

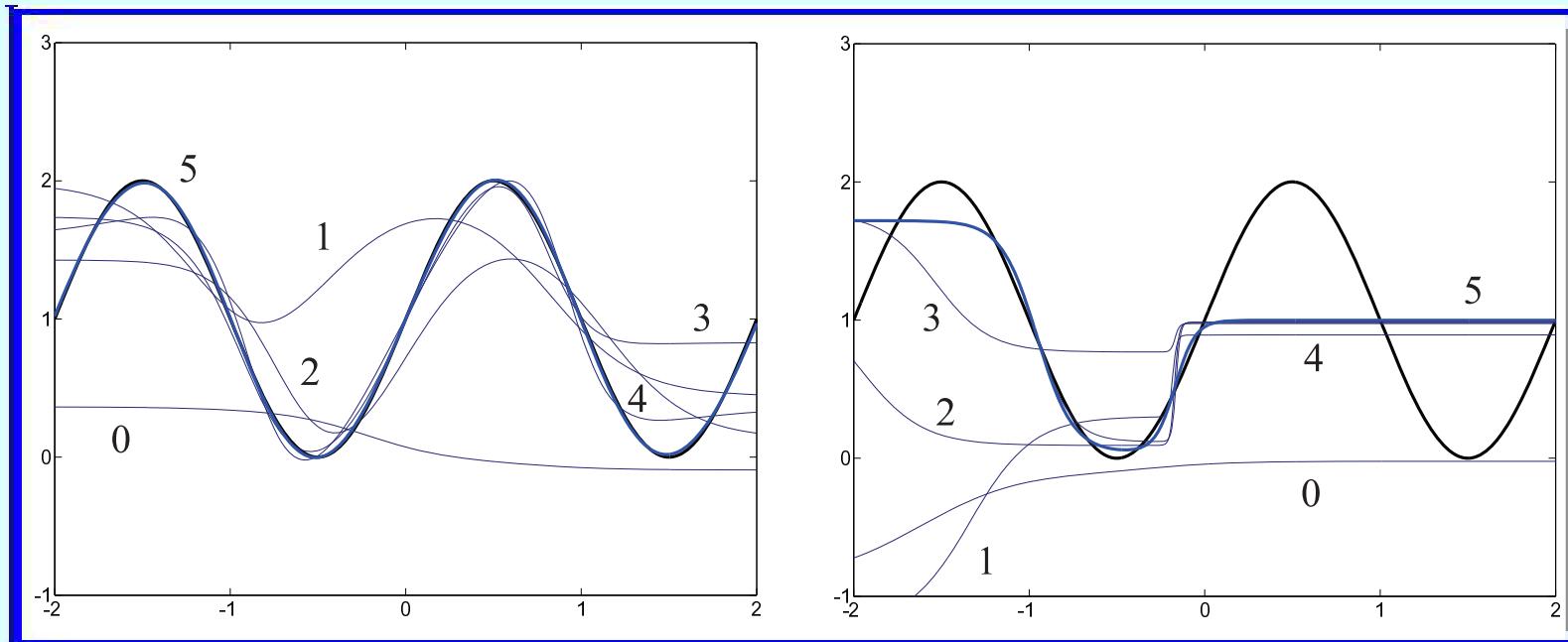
Пронумерованные кривые — последовательность состояний процесса обучения сети (числа рядом с кривыми — **НЕ** номера итераций процесса).

Завершающее состояние процесса (с номером 5) — утолщенная синяя линия.

Источник: *Hagan M. T., Demuth H.B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.13, p.11-21).

Применение сетей прямого распространения (IX)

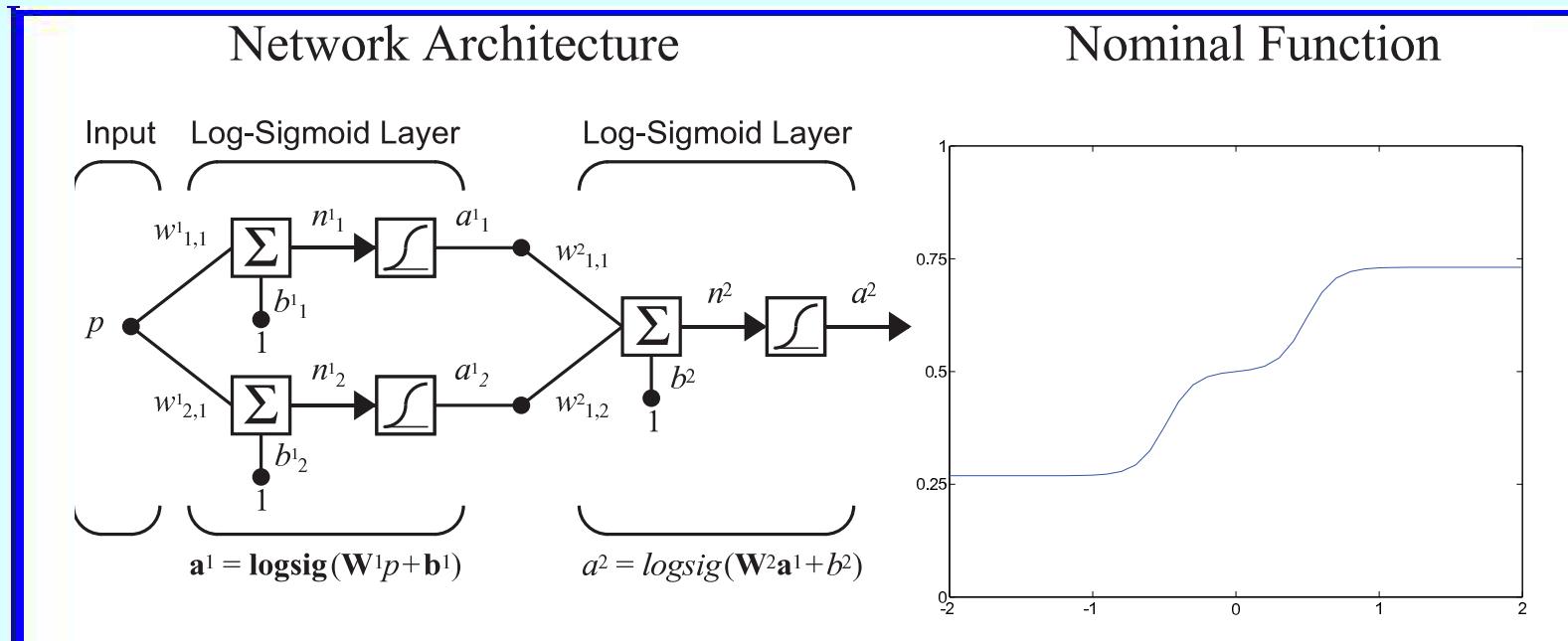
Сходимость процесса обучения сети – 3



Разница между процессами поиска минимума функции ошибки сети, изображенными на левом и правом рисунках, состоит **только** в начальных условиях (стартовые значения весов сети).

Применение сетей прямого распространения (X)

Сходимость процесса обучения сети – 4



$$w_{1,1}^1 = 10, \quad w_{2,1}^1 = 10, \quad b_1^1 = -5, \quad b_2^1 = 5$$

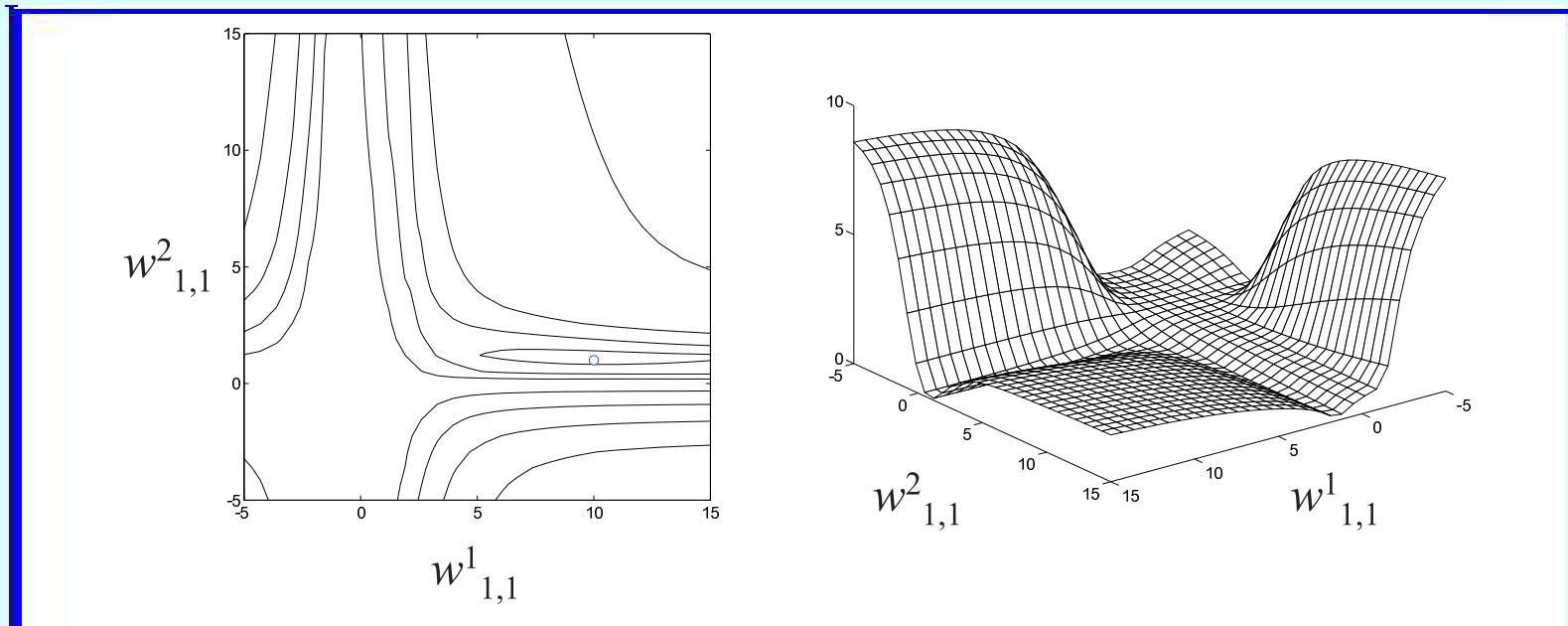
$$w_{1,1}^2 = 1, \quad w_{1,2}^1 = 1, \quad b^2 = -1$$

$$p = -2.0, -1.9, -1.8, \dots, 1.8, 1.9, 2.0$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figures 12.1 and 12.2, pp.12-3, 12-4).

Применение сетей прямого распространения (XI)

Сходимость процесса обучения сети – 5



$E = E(w_{1,1}^1, w_{1,1}^2); \quad w_{2,1}^1, w_{1,2}^2, b_1^1, b_2^1, b^2$ — оптимальные значения

$$E_{min}(w_{1,1}^1, w_{1,1}^2) = E(10, 1) = 0$$

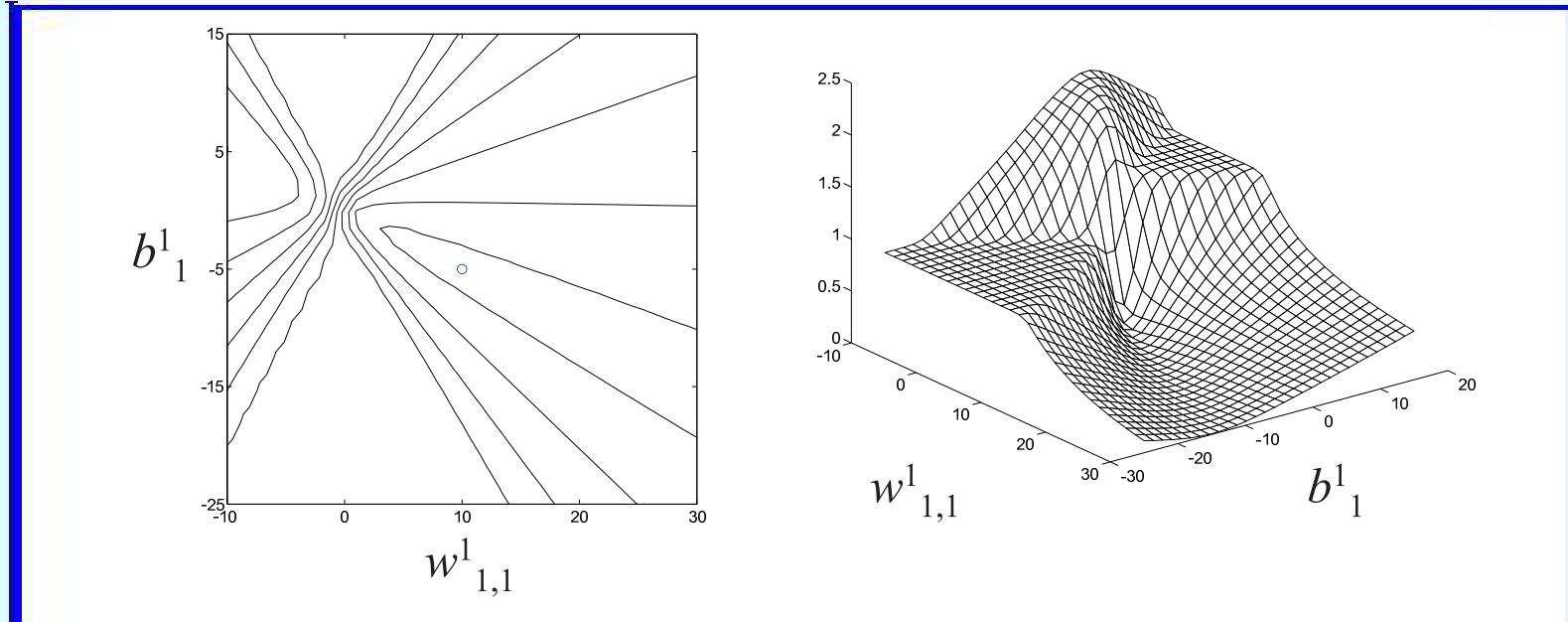
Точка **(10, 1)** — глобальный минимум функции ошибки $E(w_{1,1}^1, w_{1,1}^2)$

Точка **(0.88, 38.6)** — локальный минимум функции ошибки $E(w_{1,1}^1, w_{1,1}^2)$

Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figure 12.3, p.12-5).

Применение сетей прямого распространения (XII)

Сходимость процесса обучения сети – 6



$E = E(w_{1,1}^1, b_1^1); \quad w_{2,1}^1, w_{1,1}^2, w_{1,2}^2, b_2^1, b_2^2$ — оптимальные значения

$$E_{min}(w_{1,1}^1, b_1^1) = E(10, -5) = 0$$

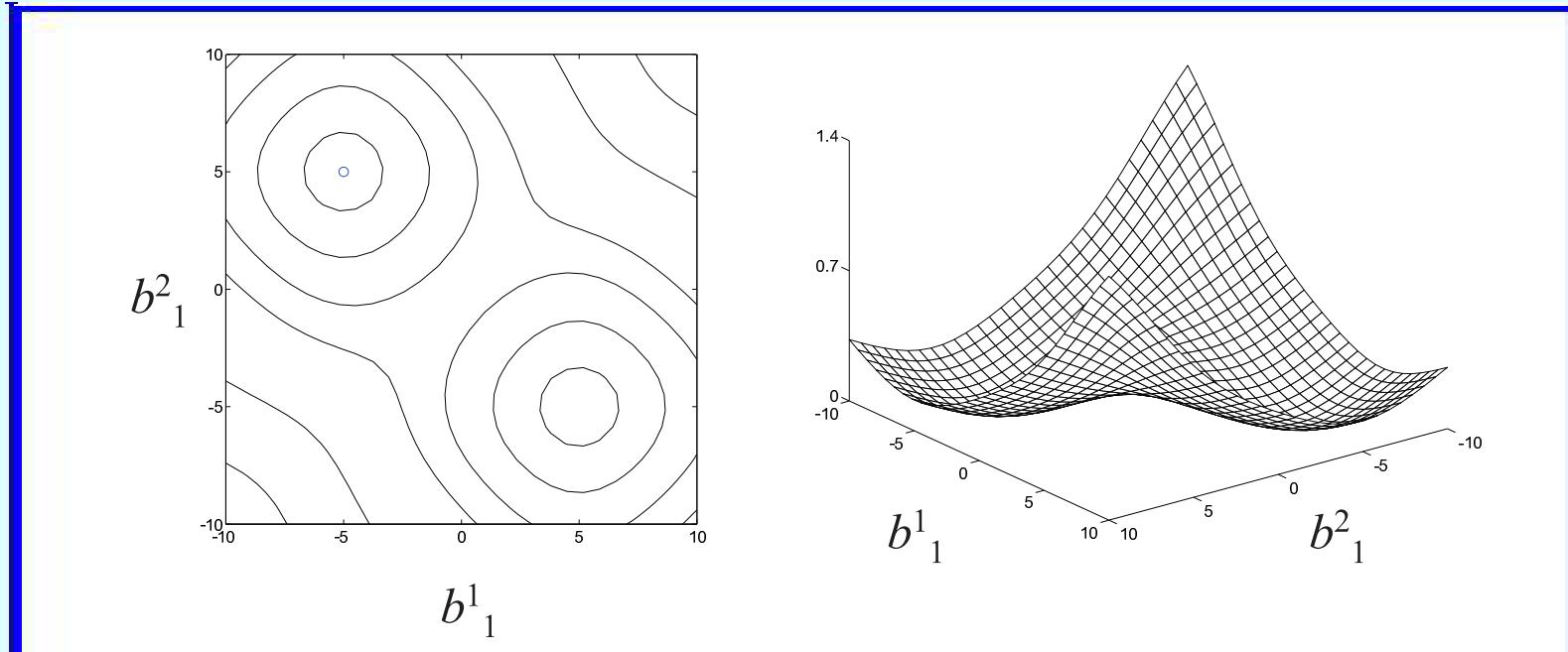
Точка $(10, -5)$ — **минимум** функции ошибки $E(w_{1,1}^1, b_1^1)$

Точка $(0, -10)$ — **градиент** функции ошибки ≈ 0 , наискорейший спуск не работает.

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figure 12.4, p.12-6).

Применение сетей прямого распространения (XIII)

Сходимость процесса обучения сети – 7



$E = E(b_1^1, b_1^2); \quad w_{1,1}^1, w_{2,1}^1, w_{1,1}^2, w_{1,2}^2, b^2$ — оптимальные значения

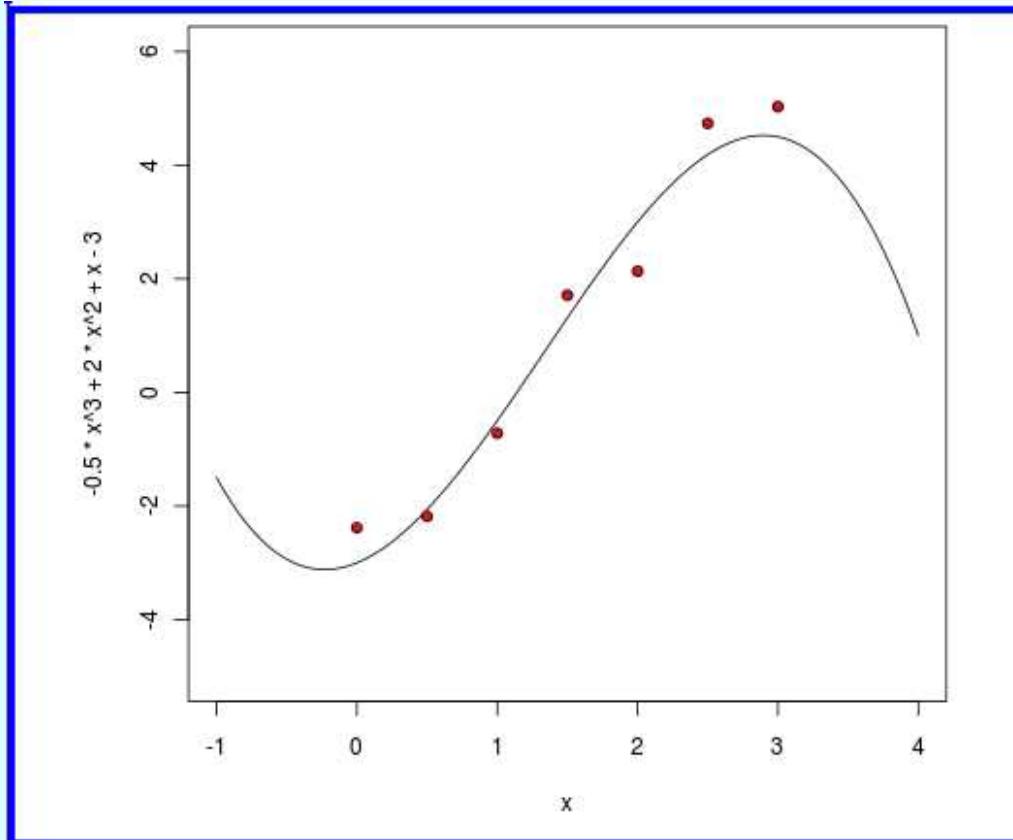
$$E_{min}(b_1^1, b_1^2) = E(5, -5) = E(-5, 5) = 0$$

Точки $(5, -5)$ и $(-5, 5)$ — локальные минимумы функции ошибки $E(b_1^1, b_1^2)$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figure 12.5, p.12-7).

Применение сетей прямого распространения (XIV)

Эффект переобученности – 1

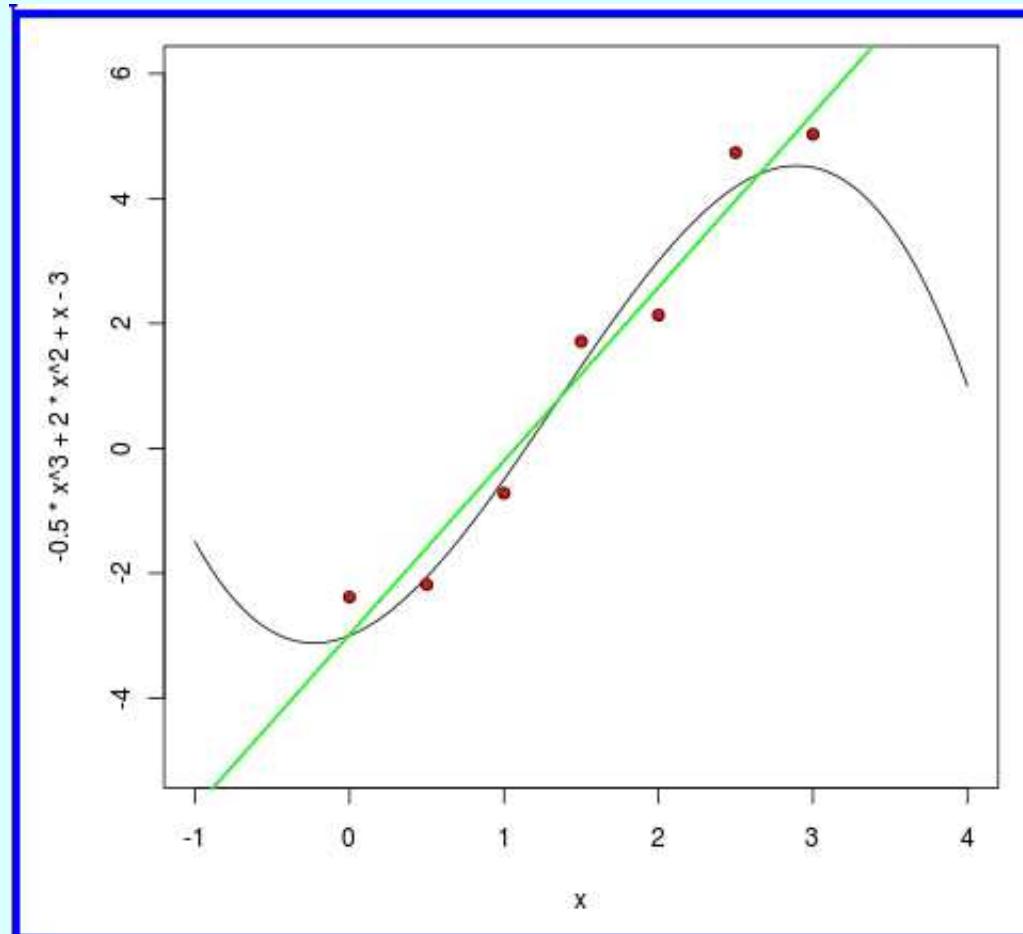


$$y(x) = -\frac{1}{2}x^3 + 2x^2 + x - 3, \text{ найти } p(x) \Rightarrow \min_x \sum_{i=1}^N (y_n - p(x_n))^2$$

Источник: [Николенко С.](#) Рекомендательные системы: оверфиттинг и регуляризация
<http://habrahabr.ru/company/surfingbird/blog/143455/#habracut>

Применение сетей прямого распространения (XV)

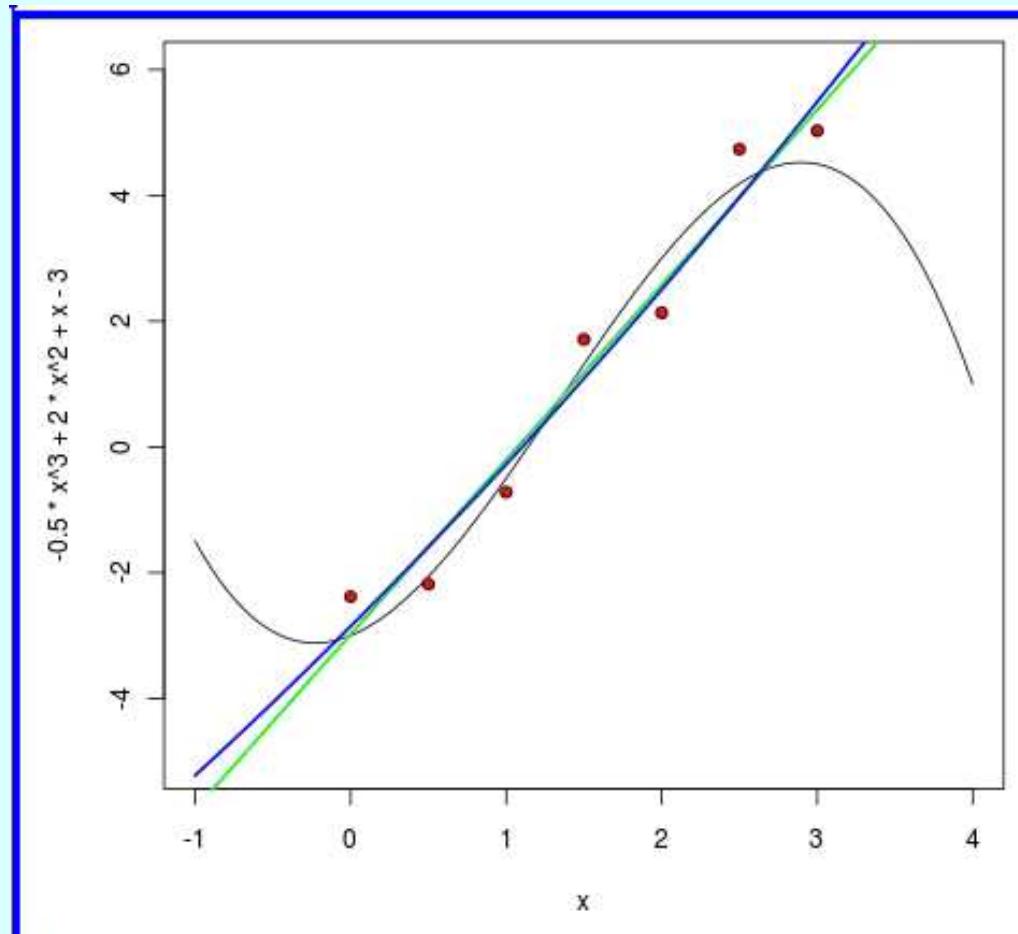
Эффект переобученности – 2



$p(x)$ – многочлен **первой** степени

Применение сетей прямого распространения (XVI)

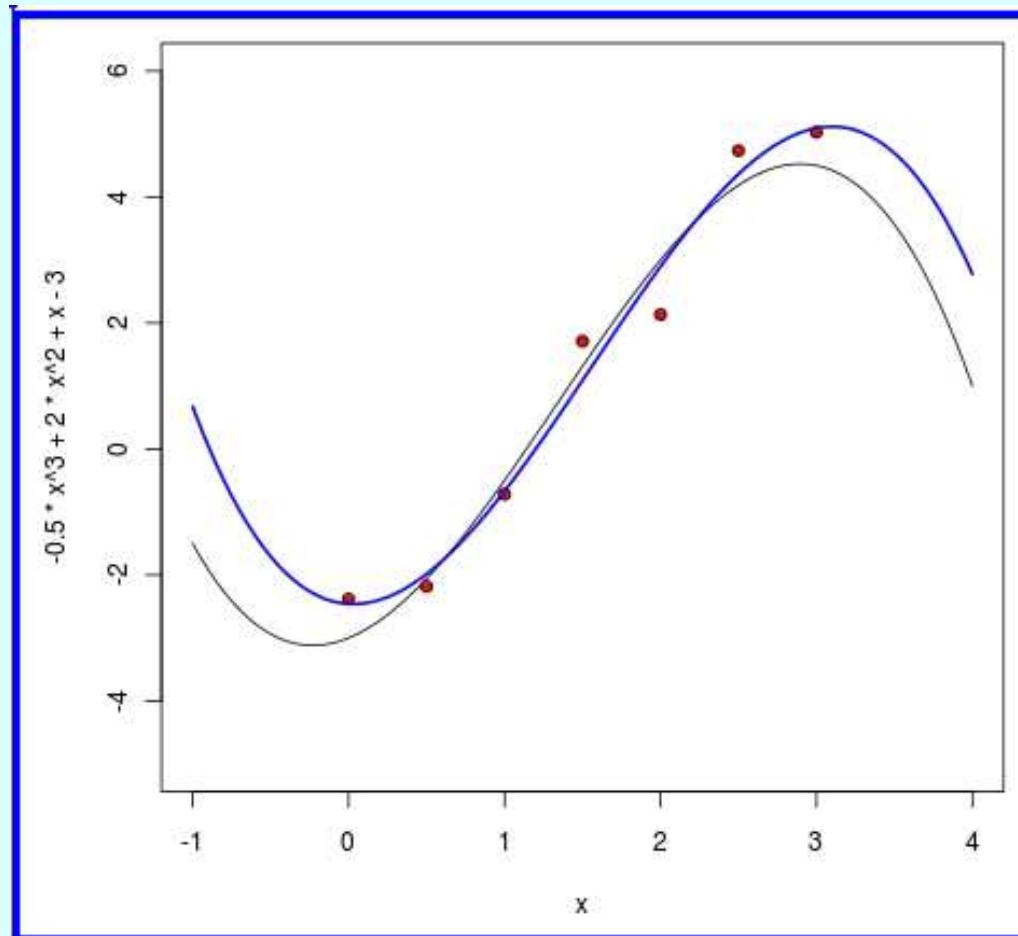
Эффект переобученности – 3



$p(x)$ – многочлен **второй** степени

Применение сетей прямого распространения (XVII)

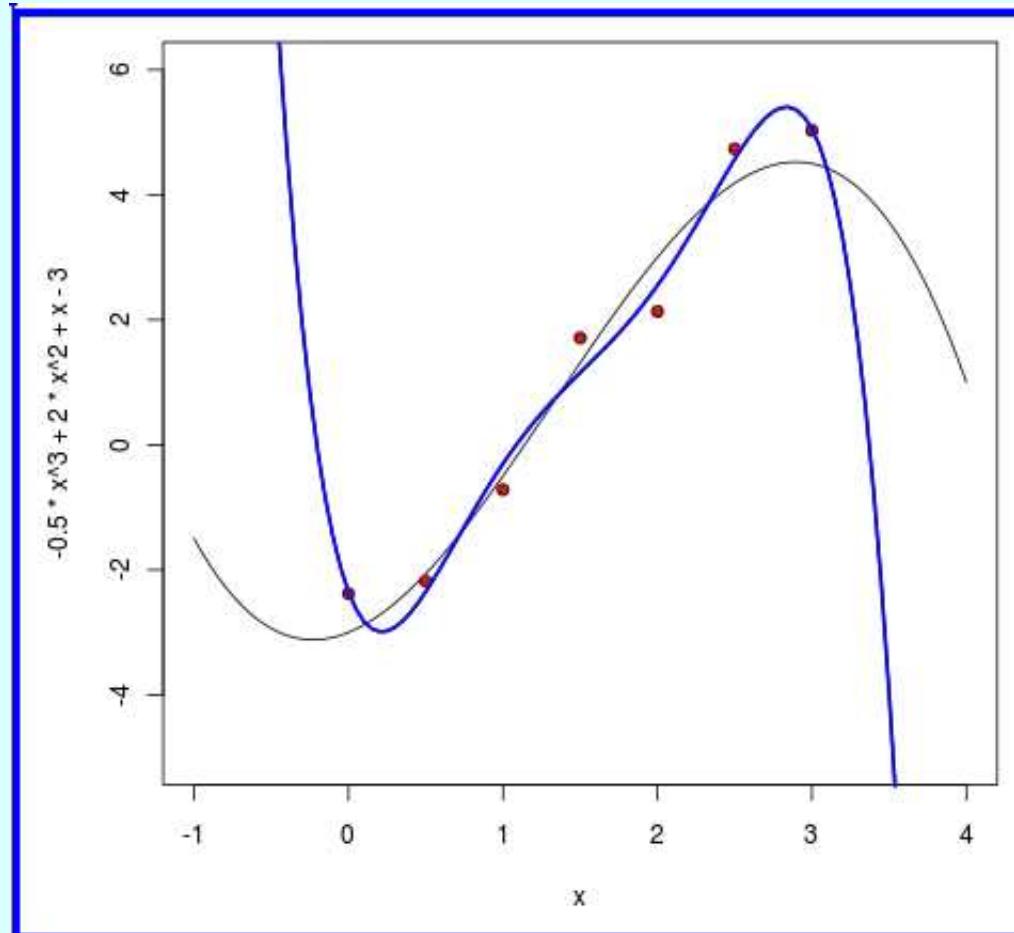
Эффект переобученности – 4



$p(x)$ – многочлен **третьей** степени

Применение сетей прямого распространения (XVIII)

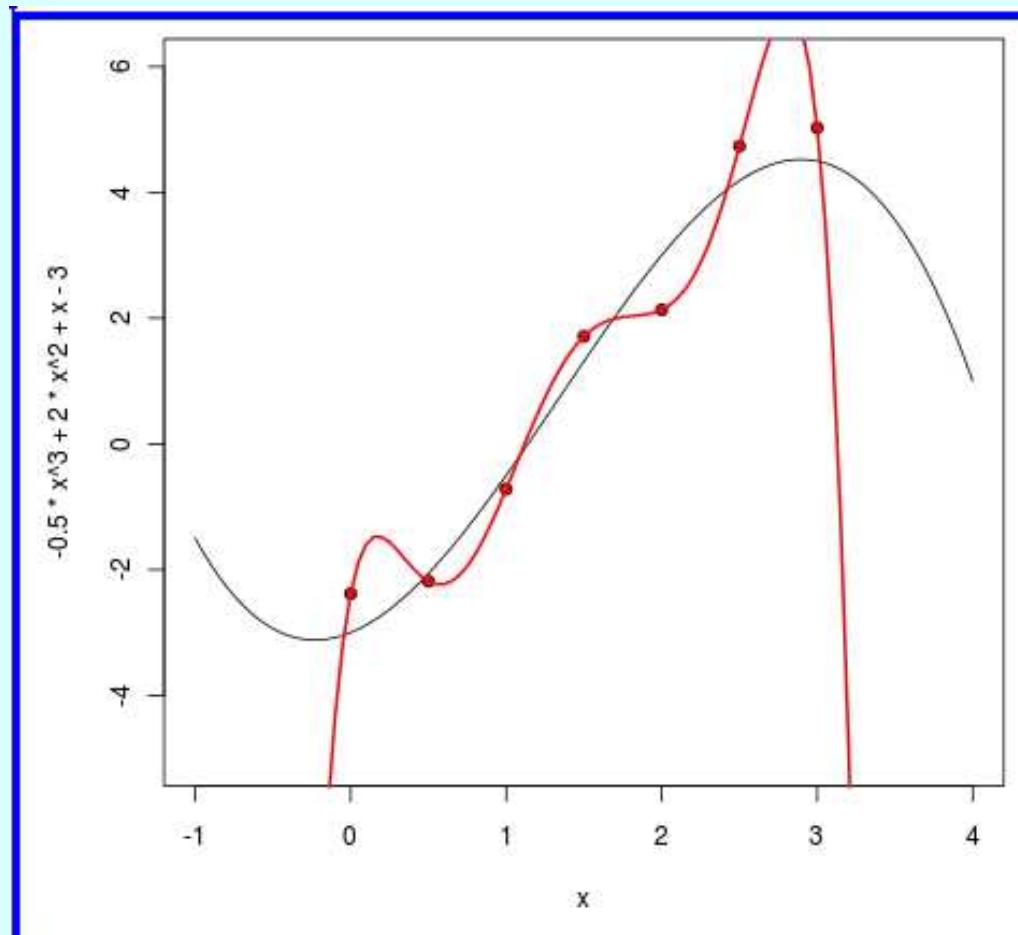
Эффект переобученности – 5



$p(x)$ — многочлен **пятой** степени

Применение сетей прямого распространения (XIX)

Эффект переобученности – 6

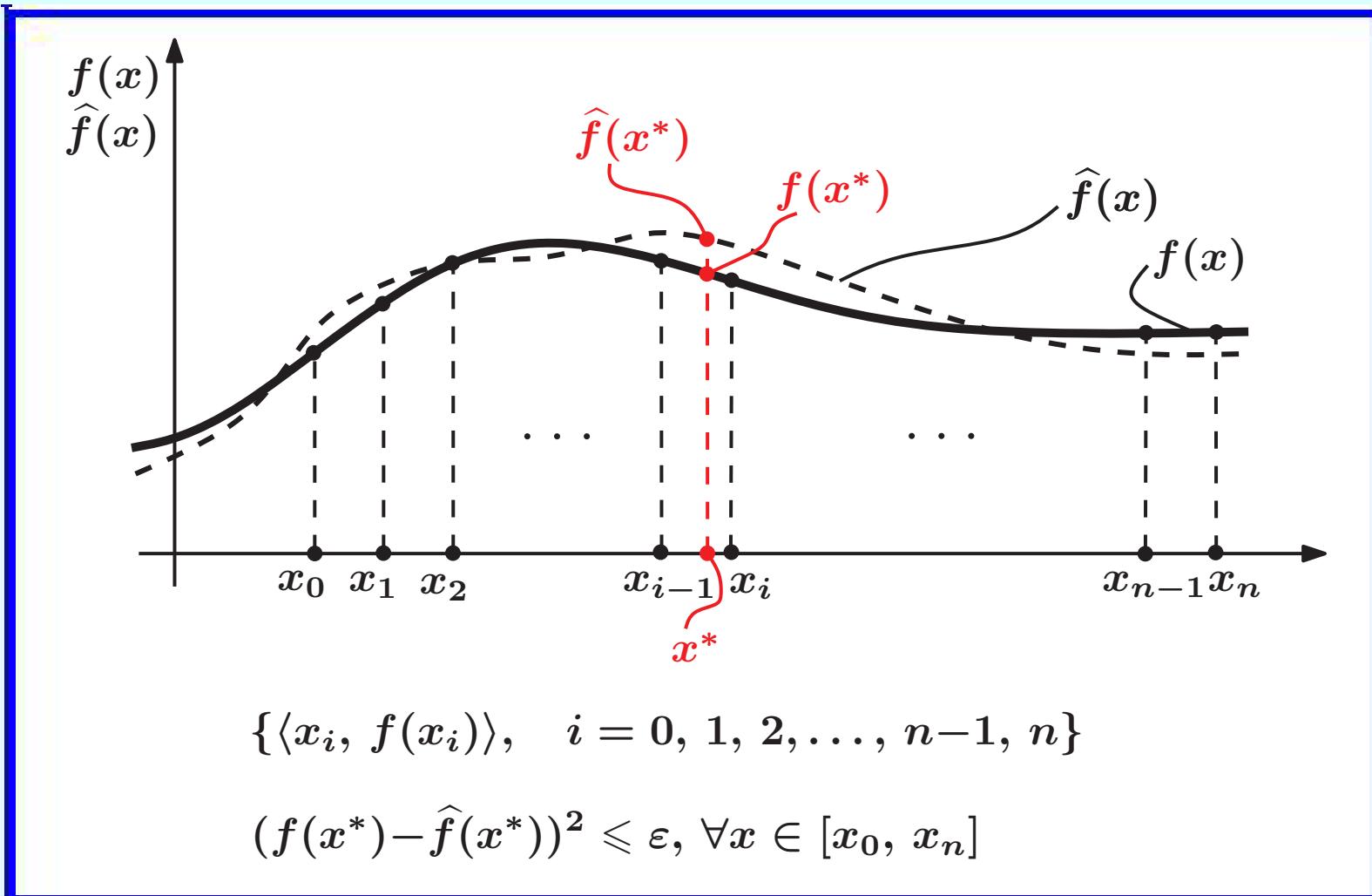


$p(x)$ – многочлен **шестой** степени

Переобученность (оверфиттинг, от англ. overfitting)

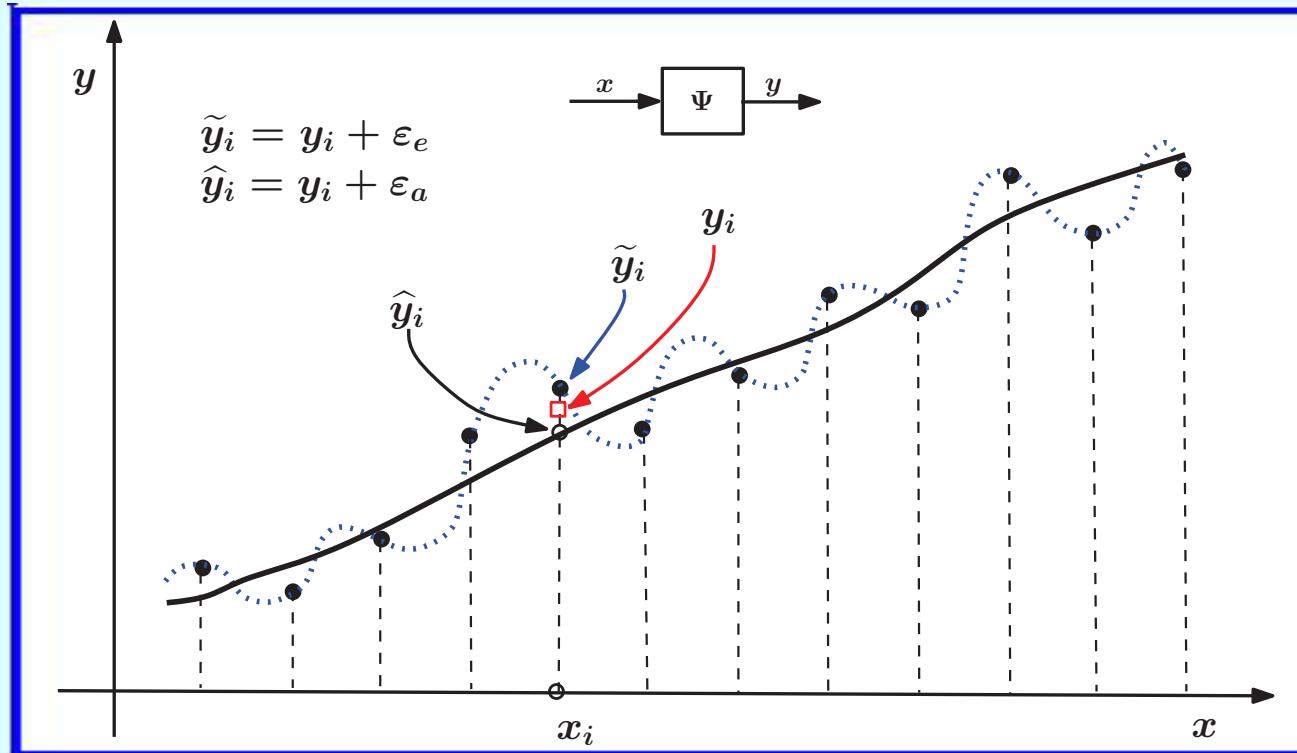
Применение сетей прямого распространения (XX)

Обобщающие свойства сетей – 1



Применение сетей прямого распространения (ХХI)

Обобщающие свойства сетей – 2



y_i — истинное значение функции $y = \Psi(x)$ при x_i

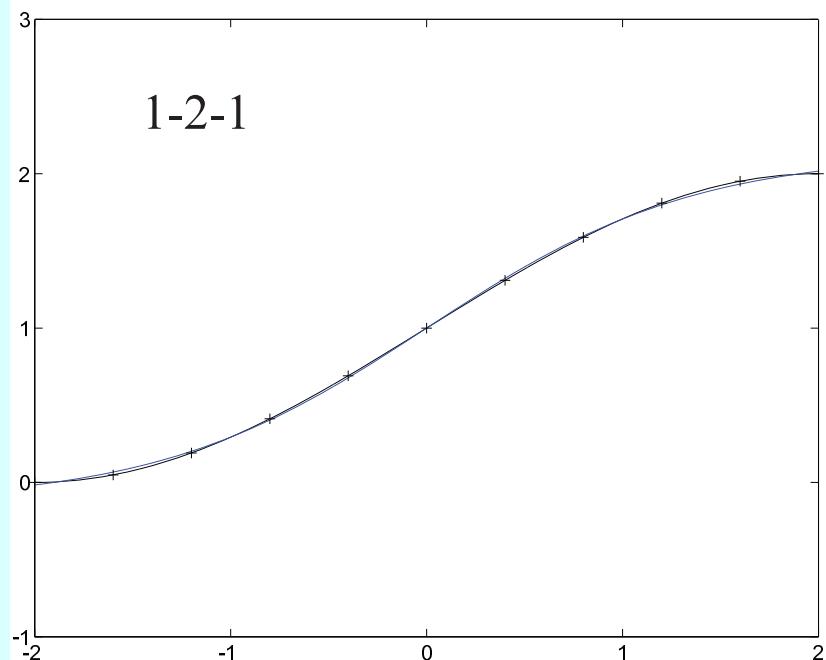
\tilde{y}_i — значение функции $y = \Psi(x)$, полученное из эксперимента при x_i

\hat{y}_i — значение функции $y = \hat{\Psi}(x)$, аппроксимирующей функцию $y = \Psi(x)$ при x_i

$\varepsilon_e, \varepsilon_a$ — погрешности измерения и аппроксимации, соответственно

Применение сетей прямого распространения (ХII)

Обобщающие свойства сетей – 3



Влияние числа нейронов
в скрытом слое сети
(синяя линия — отклик сети)

Аппроксимируемая функция:

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right),$$

$$\mathbf{p} = (p_1, p_2, \dots, p_{11}) =$$

$$= (-2, -1.6, -1.2, \dots, 1.6, 2)$$

$$\mathbf{t} = (t_1, t_2, \dots, t_{11}) =$$

$$= (g(p_1), g(p_2), \dots, g(p_{11}))$$

Сеть со структурой 1-2-1

7 настраиваемых параметров:

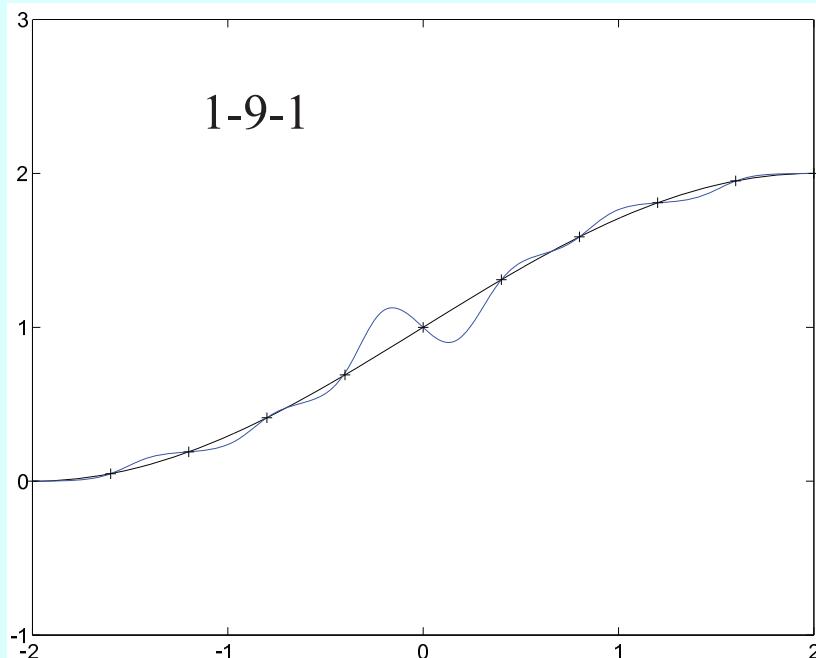
4 веса

3 смещения

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.14, p.11-22).

Применение сетей прямого распространения (ХХIII)

Обобщающие свойства сетей – 4



Влияние числа нейронов
в скрытом слое сети
(синяя линия — отклик сети)

Аппроксимируемая функция:

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right),$$

$$\mathbf{p} = (p_1, p_2, \dots, p_{11}) =$$

$$= (-2, -1.6, -1.2, \dots, 1.6, 2)$$

$$\mathbf{t} = (t_1, t_2, \dots, t_{11}) =$$

$$= (g(p_1), g(p_2), \dots, g(p_{11}))$$

Сеть со структурой 1-9-1

28 настраиваемых параметров:

18 весов

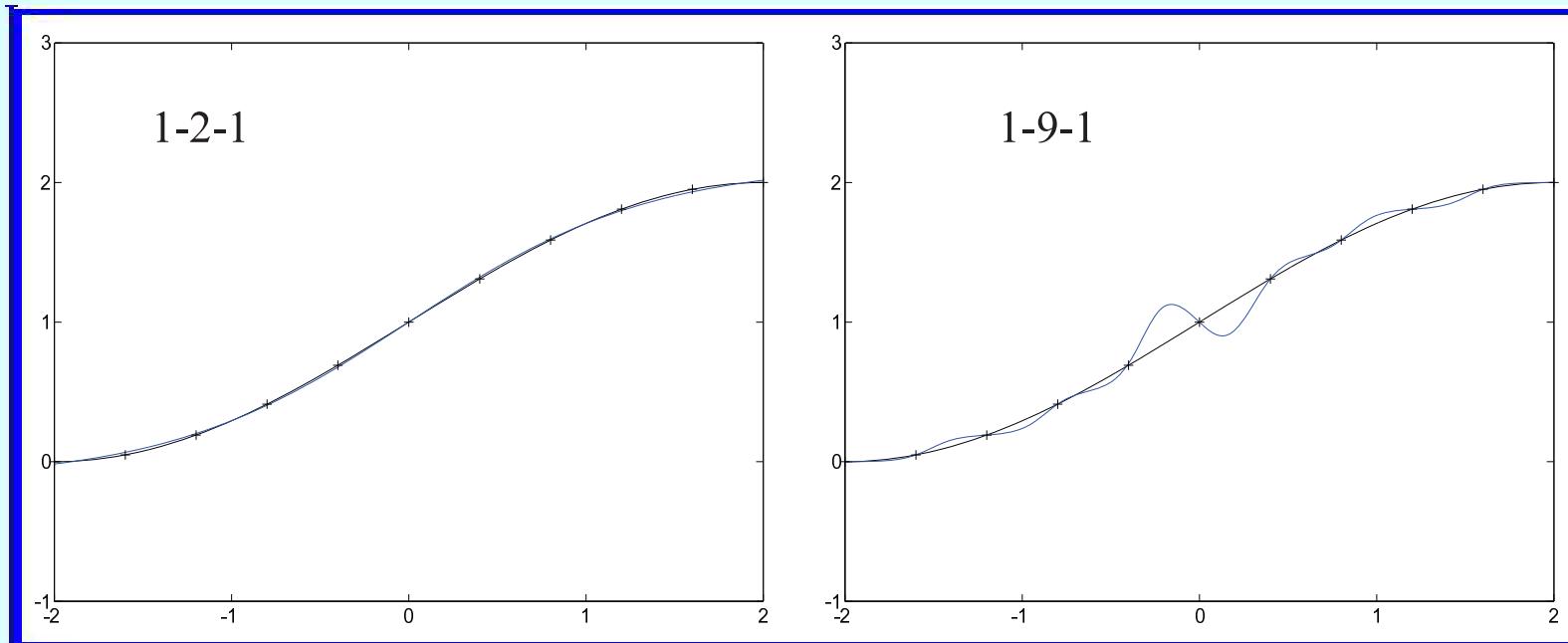
10 смещений

Источник: *Hagan M. T., Demuth H.B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 11, Figure 11.15, p.11-22).

Применение сетей прямого распространения (XXIV)

Обобщающие свойства сетей – 5

Влияние числа нейронов в скрытом слое сети



Сеть со структурой 1-2-1

7 настраиваемых параметров: 4 веса + 3 смещения

Сеть со структурой 1-9-1

28 настраиваемых параметров: 18 весов + 10 смещений

Обучающий набор: 11 примеров — достаточно для структуры 1-2-1, слишком мало для 1-9-1

Применение сетей прямого распространения (ХХV)

Проблемы, возникающие при обучении сетей – 1

Явления недоразмеренности/переразмеренности и недообученности/переобученности

Если ошибка моделирования не достигает требуемой величины,
то это может быть следствием:

- недообученности** сети — выполнено недостаточное число итераций обучающего алгоритма;
- недоразмеренности** сети — число элементов в скрытых слоях сети недостаточно.

В случае **недообученности** следует **продолжить** процесс обучения, ошибка обучения **будет снижаться**.

В случае **недоразмеренности** сеть не располагает требуемой «аппроксимирующей силой» и продолжение процесса обучения **бесполезно**, поскольку ошибка моделирования **снижаться уже не будет**.

Применение сетей прямого распространения (XXVI)

Проблемы, возникающие при обучении сетей – 2

Явления недоразмеренности/переразмеренности и недообученности/переобученности

Слишком **долгое обучение** сети и слишком **большое число элементов** в ней также **ухудшают качество** получаемой нейросетевой модели.

Сеть в этом случае становится:

- переразмеренной** — число нейронов в ее скрытых слоях слишком велико;
- переобученной** — число эпох в процессе ее обучения слишком велико.

Переразмеренная сеть обладает слишком большим числом «степеней свободы», поэтому **обобщающие свойства** ее неудовлетворительны.

Переобученной сеть становится, если в процессе ее обучения превышено некоторое **критическое число эпох** N_r^* . Переобученность сети проявляется в том, что она начинает воспроизводить такие особенности обучающего набора, как, например, **флуктуации**, вызванные измерительными шумами.

Применение сетей прямого распространения (XXVII)

Проблемы, возникающие при обучении сетей – 3

Критическое число эпох (1)

Поиск критического числа эпох N_r^* — на примере **двухслойных сетей** с числом нейронов в скрытом слое $N_H = 8, 16$ и 24 .

В этих сетях нейроны **в скрытом слое** используют гауссовскую активационную функцию, а нейроны **в выходном слое** — линейную активационную функцию.

Критическое число эпох N_r^* определяется **сопоставлением ошибки** сети на обучающем и тестовом наборе, которое можно начинать **лишь с некоторой эпохи** \widetilde{N}_r , номер которой тем больше, чем большее число нейронов содержится в скрытом слое сети.

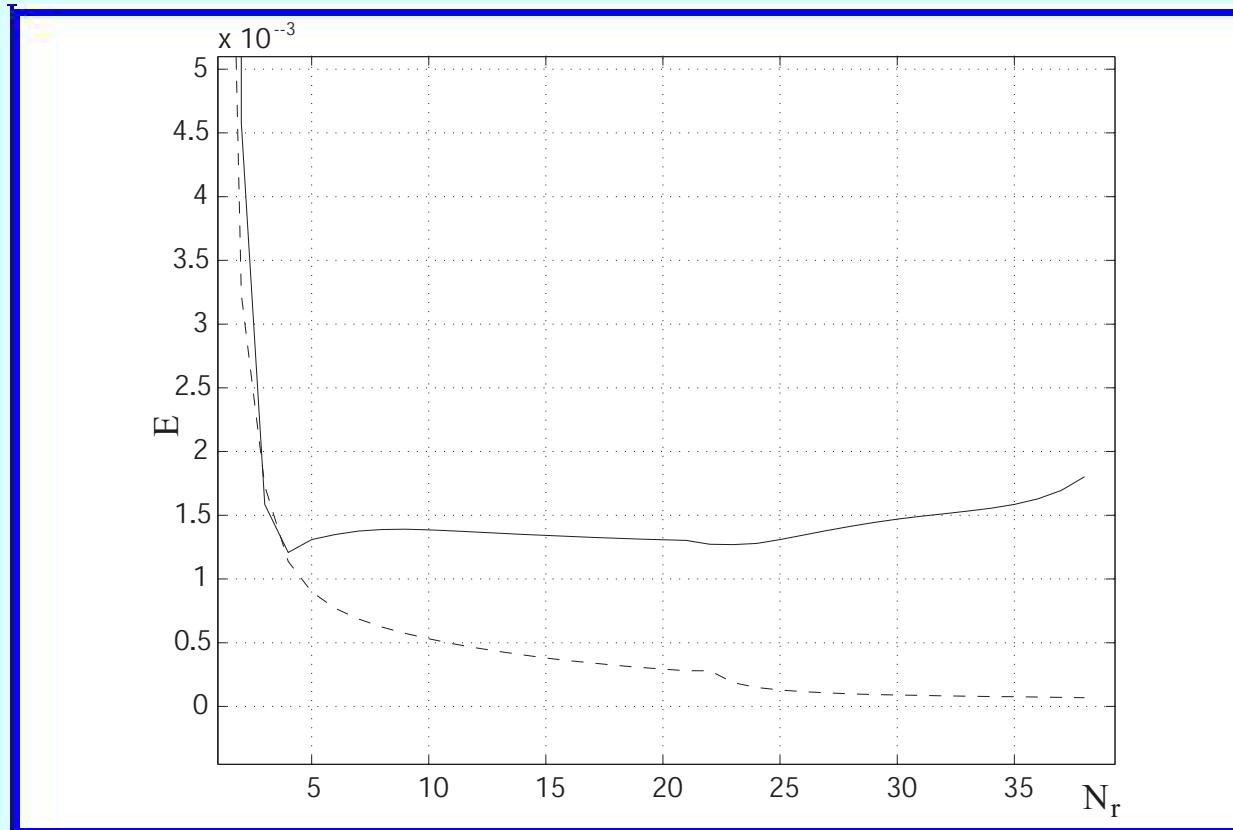
В рассматриваемых примерах:

- при $N_H = 8$ — практически сразу,
- при $N_H = 16$ — примерно с $\widetilde{N}_r = 8$,
- при $N_H = 24$ — примерно с $\widetilde{N}_r = 12$.

При числе эпох, меньшем \widetilde{N}_r , **степень обученности** сети еще слишком низка и проверка ее на тестовом наборе дает **неадекватные результаты**.

Применение сетей прямого распространения (XXVIII)

Проблемы, возникающие при обучении сетей – 4 Критическое число эпох (2)



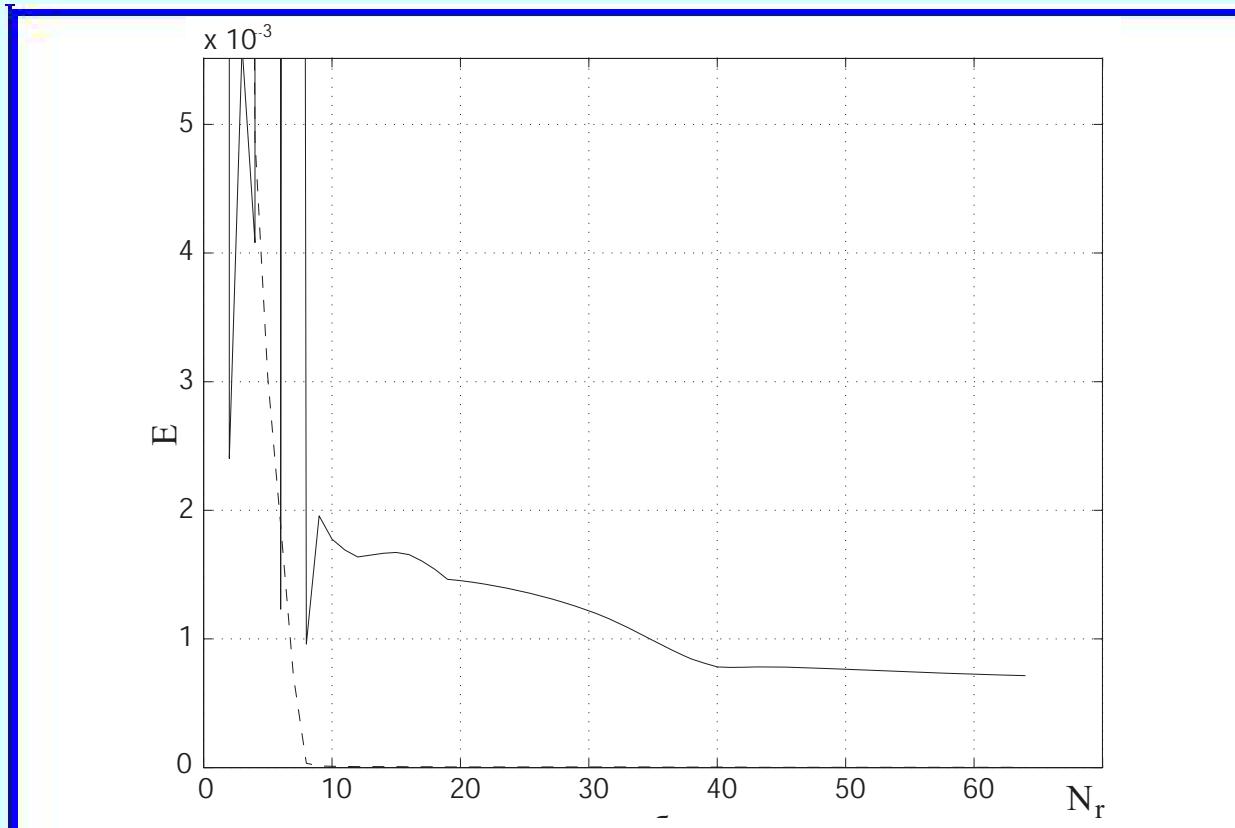
8 нейронов в единственном скрытом слое

Ошибка сети E на тестовом наборе (сплошная линия)
и на обучающем наборе (пунктирная линия)

N_r — число эпох

Применение сетей прямого распространения (XXIX)

Проблемы, возникающие при обучении сетей – 5 Критическое число эпох (3)



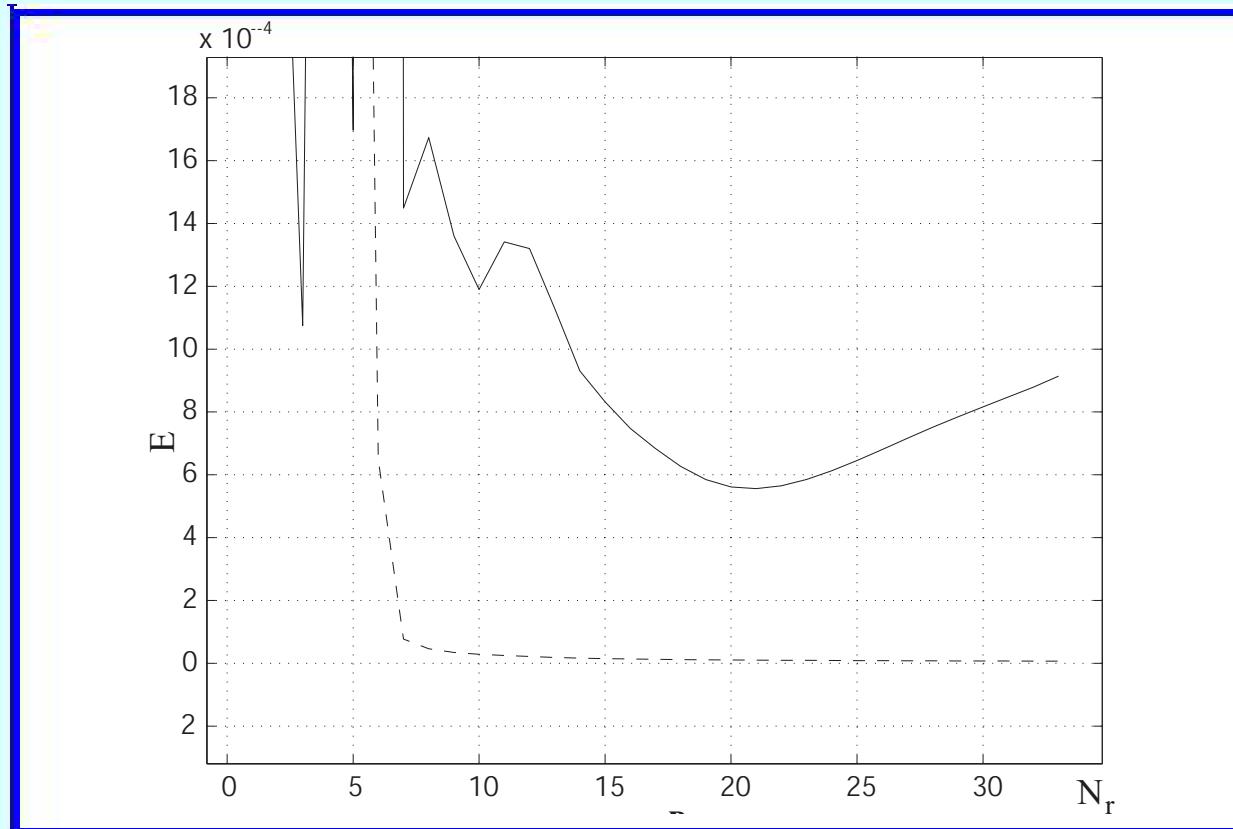
16 нейронов в единственном скрытом слое

Ошибка сети E на тестовом наборе (сплошная линия)
и на обучающем наборе (пунктирная линия)

N_r — число эпох

Применение сетей прямого распространения (XXX)

Проблемы, возникающие при обучении сетей – 6 Критическое число эпох (4)



24 нейрона в единственном скрытом слое

Ошибка сети E на тестовом наборе (сплошная линия)
и на обучающем наборе (пунктирная линия)

N_r — число эпох

Применение сетей прямого распространения (XXXI)

Проблемы, возникающие при обучении сетей – 7

Критическое число эпох (5)

С ростом числа N_H элементов в скрытом слое сети:

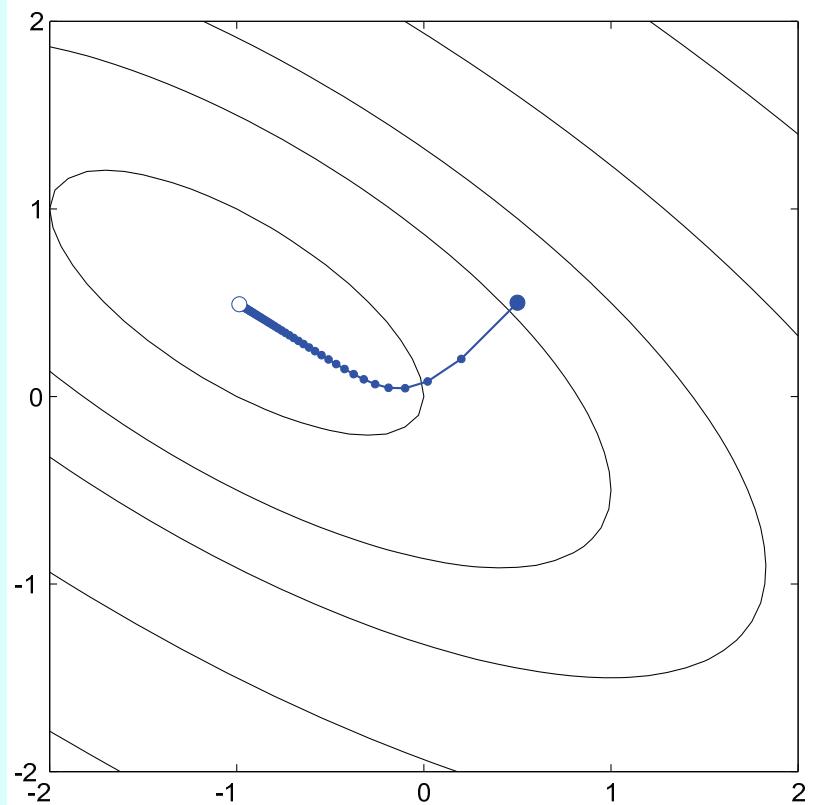
- повышается потенциально достижимая **точность аппроксимации**, т. е. снижается ошибка моделирования $\mathcal{E}(\cdot)$;
- снижается **критическое значение** N_r^* числа эпох, после которого сеть становится **переобученной**.

Чтобы избежать переобученности при увеличении числа нейронов в скрытых слоях сети, требуется:

- раньше завершать процесс обучения сети, *или*
- увеличить число примеров в обучающем наборе.

Применение сетей прямого распространения (XXXII)

Недостатки исходного алгоритма ВР – 1



Алгоритм обратного распространения ошибки в его исходном варианте (**SDBP** – Steepest Descent Back-Propagation) основан на оптимизационной схеме **наискорейшего спуска**.

Основной недостаток алгоритма SDBP – **очень медленная сходимость**, для получения решения может потребоваться до нескольких десятков тысяч итераций.

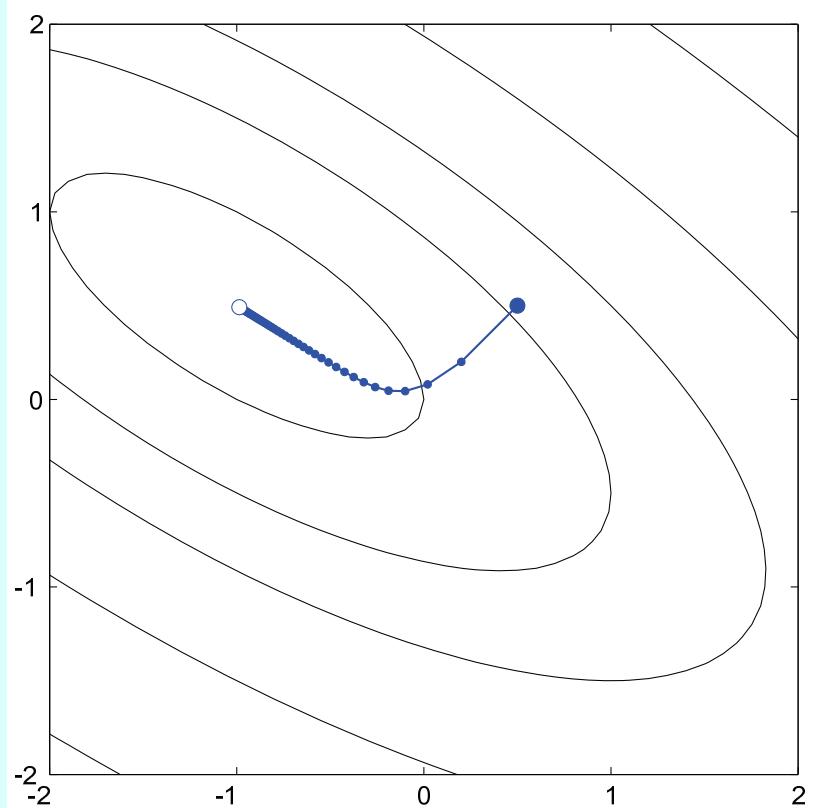
Алгоритм **SDBP** – обобщение алгоритма **LMS**, который хорошо справляется с задачей обучения **однослоиной линейной сети**, квадратичная функция ошибки которой имеет **единственный минимум**.

Многослойные нелинейные сети – многоэкстремальная функция ошибки со сложным рельефом, на котором алгоритм SDBP работает неудовлетворительно.

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.1, p.9-5).

Применение сетей прямого распространения (XXXIII)

Недостатки исходного алгоритма ВР – 2



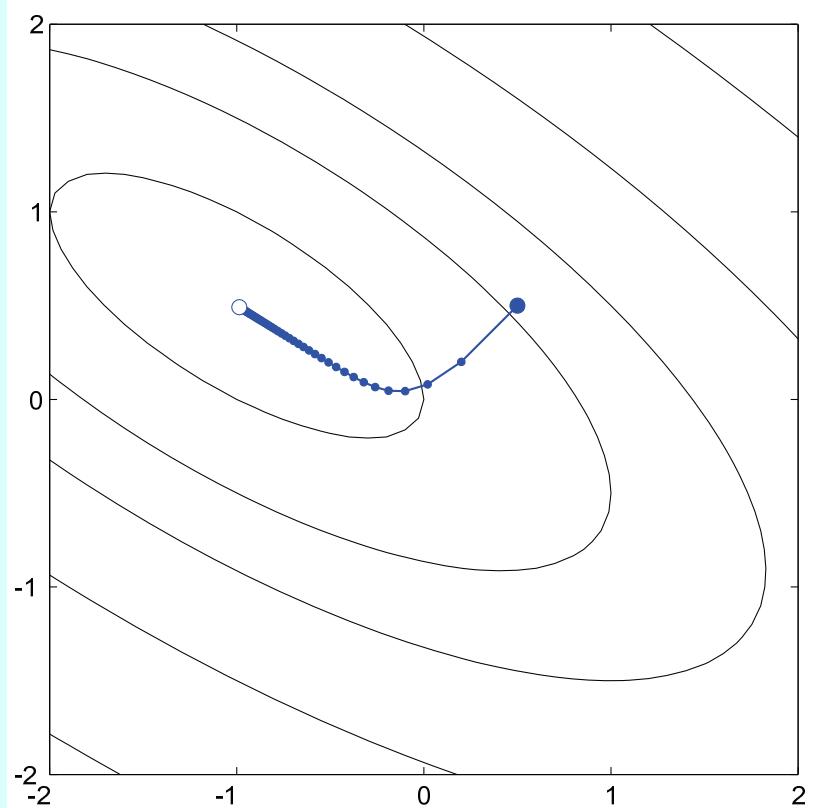
**Возможные пути устранения
недостатков SDBP:**

- **эвристические улучшения** алгоритма SDBP, позволяющие повысить эффективность поиска решения **без замены** используемой оптимизационной схемы (наискорейший спуск);
- использование вместо наискорейшего спуска **другой оптимизационной схемы**.

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.1, p.9-5).

Применение сетей прямого распространения (XXXIV)

Недостатки исходного алгоритма ВР – 3



Эвристические улучшения алгоритма SDBP:

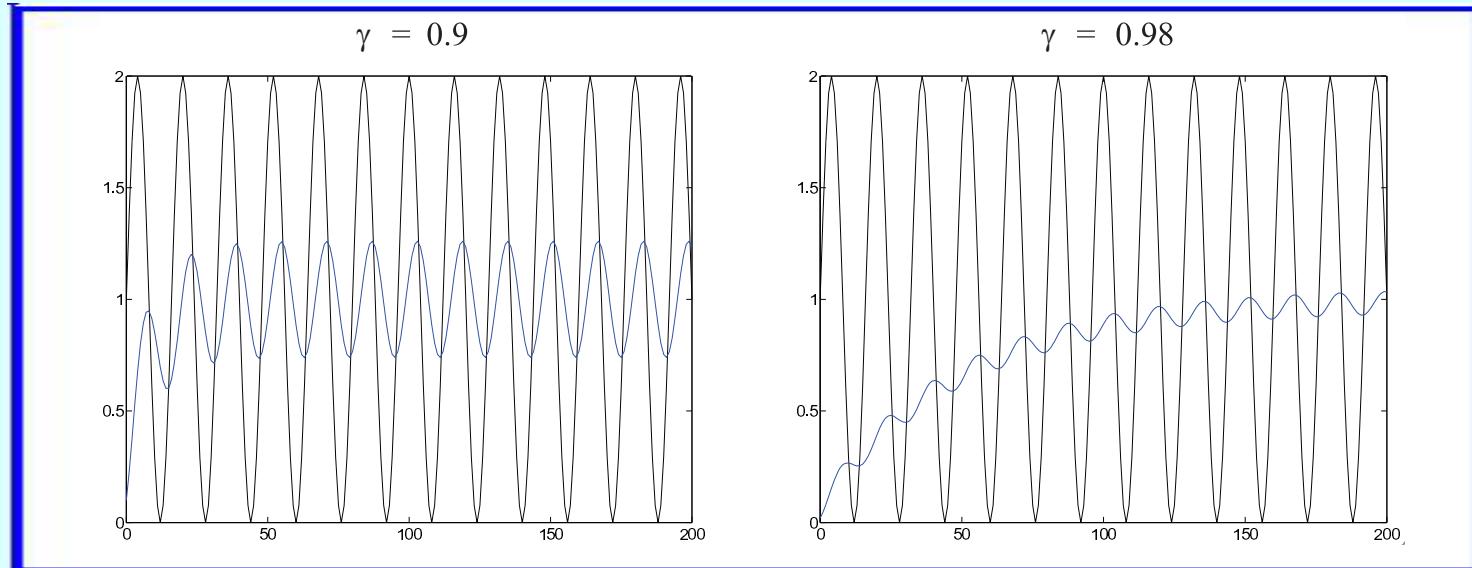
- **метод моментов** — введение в схему SDBP элемента, подобного **фильтру низких частот**, для уменьшения «раскачки» траектории поиска решения;
- **переменная скорость обучения**, подстраиваемая с учетом изменения **кривизны поверхности**, задаваемой функцией ошибки, в текущей зоне поиска решения.

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.1, p.9-5).

Применение сетей прямого распространения (XXXV)

Недостатки исходного алгоритма ВР – 4

Метод моментов (1)



Пример фильтра:

$$y(k) = \gamma y(k - 1) + (1 - \gamma)w(k), \quad 0 \leq \gamma < 1$$

Пример сигнала на входе в фильтр:

$$w(k) = 1 + \sin\left(\frac{2\pi k}{16}\right)$$

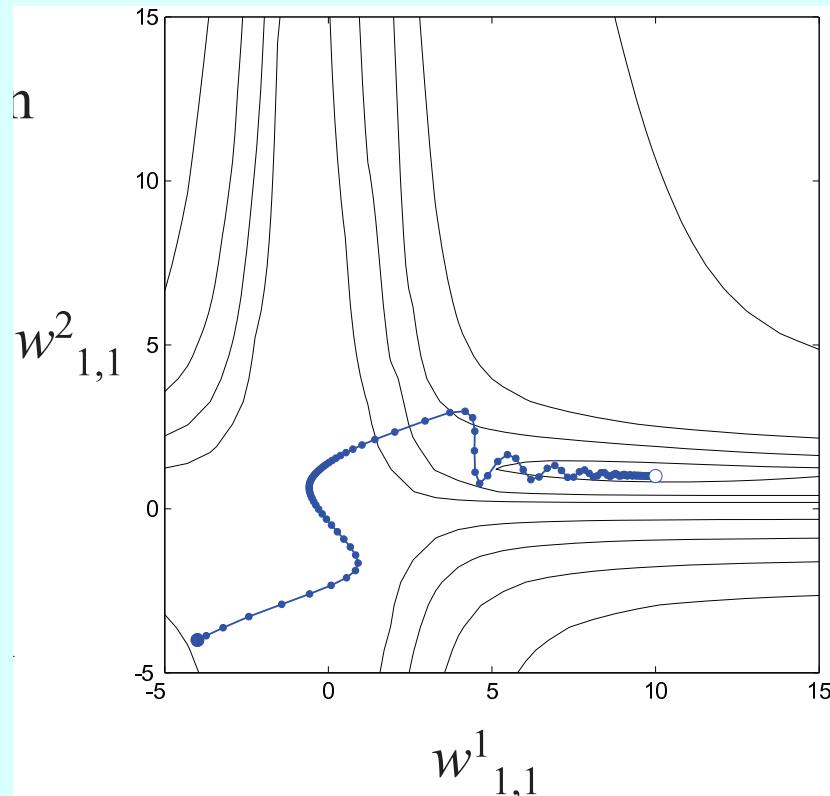
Сигнал **на выходе фильтра** (синяя линия) имеет амплитуды значительно меньшие, чем сигнал **на входе фильтра** (черная линия).

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figure 12.9, p.12-10).

Применение сетей прямого распространения (XXXVI)

Недостатки исходного алгоритма ВР – 5

Метод моментов (2)



Алгоритм SDBP:

$$\Delta \mathbf{W}^m(k) = -\alpha \delta^m (\mathbf{a}^{m-1})^T$$
$$\mathbf{b}^m(k) = -\alpha \delta^m$$

Алгоритм SDBP с моментом:

$$\Delta \mathbf{W}^m(k) = \gamma \Delta \mathbf{W}^m(k-1) -$$
$$-(1-\gamma)\alpha \delta^m (\mathbf{a}^{m-1})^T$$

$$\Delta \mathbf{b}^m(k) = \gamma \Delta \mathbf{b}^m(k-1) - (1-\gamma)\alpha \delta^m$$
$$0 \leq \gamma < 1 - \text{момент}$$

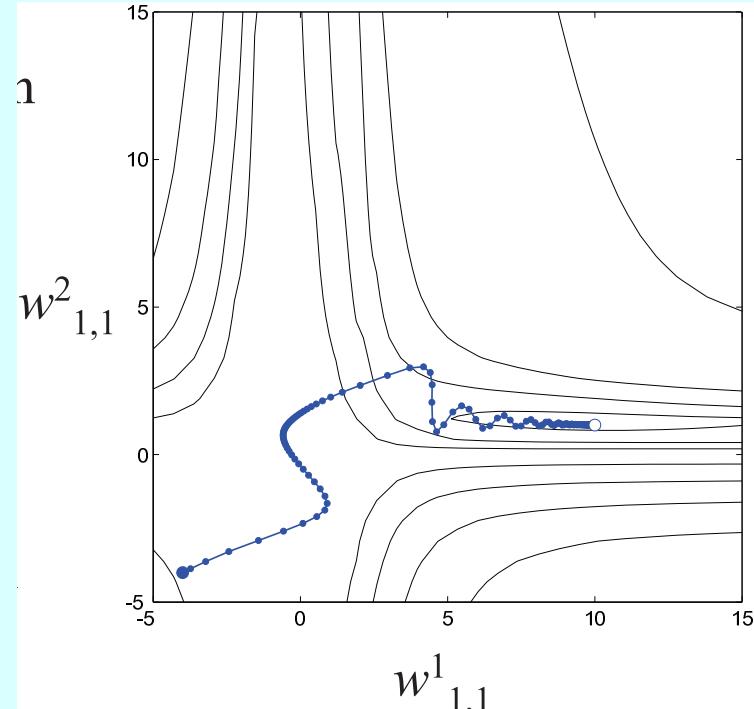
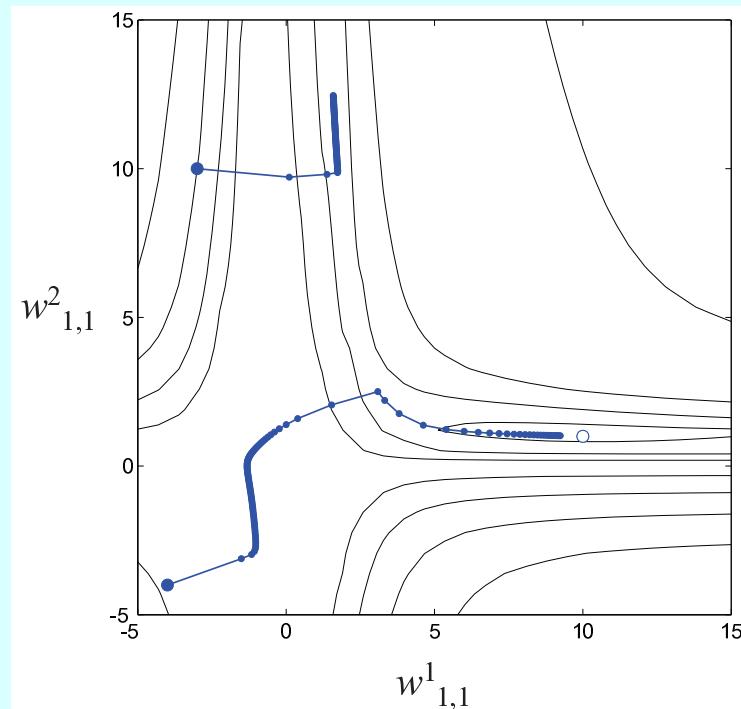
Пример на рисунке слева — для $\gamma = 0.8$

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.1, p.9-5).

Применение сетей прямого распространения (XXXVII)

Недостатки исходного алгоритма ВР – 6

Метод моментов (3)

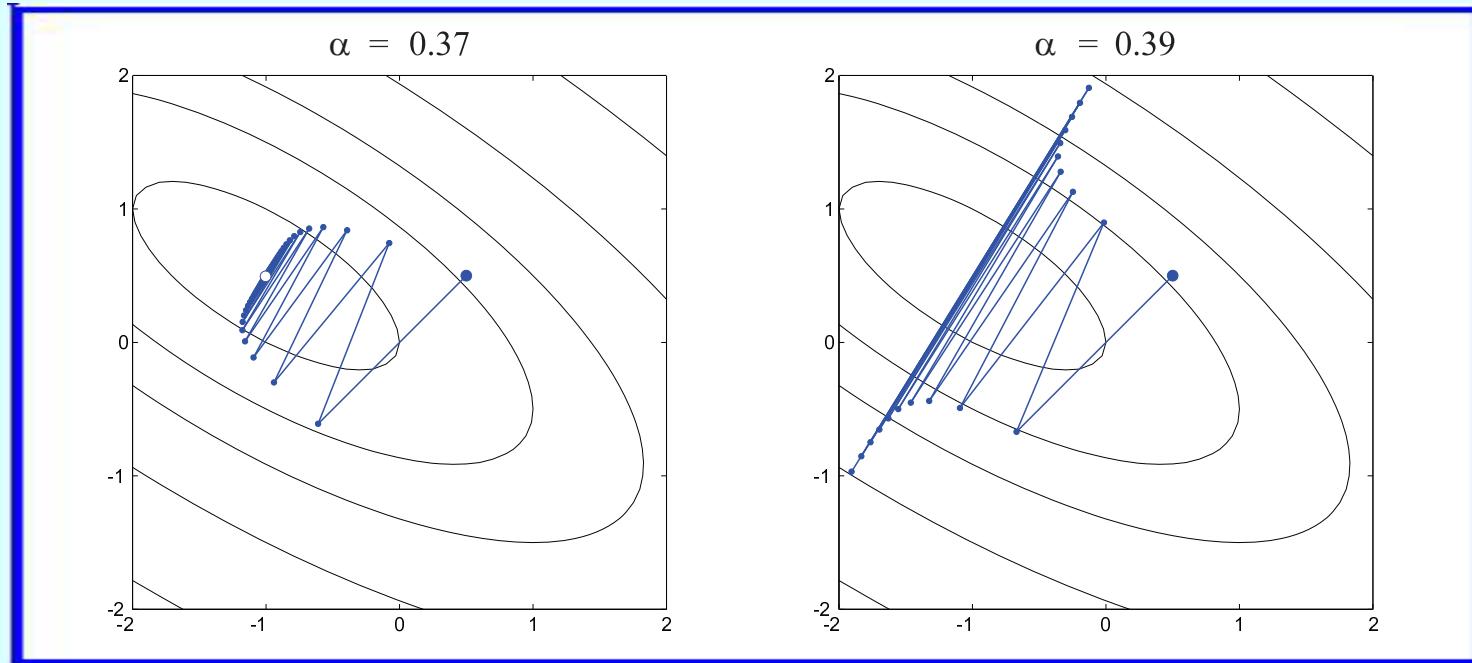


Траектория поиска решения,
реализуемая **стандартным** алгоритмом SDBP (слева)
и алгоритмом SDBP с **моментом** (справа).

Применение сетей прямого распространения (XXXVIII)

Недостатки исходного алгоритма ВР – 7

Переход к переменной скорости обучения (1)



Алгоритм SDBP с постоянной скоростью обучения

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k, \quad \alpha = \text{const}$$

Источник: *Hagan M. T., Demuth H.B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.3, p.9-7).

Применение сетей прямого распространения (XXXIX)

Недостатки исходного алгоритма ВР – 8

Переход к переменной скорости обучения (2)

Алгоритм SDBP с постоянной скоростью обучения
(условие устойчивости)

$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c, \quad \nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k = \mathbf{x} - \alpha (\mathbf{A} \mathbf{x} + \mathbf{d})$$



$$\mathbf{x}_{k+1} = [\mathbf{I} - \alpha \mathbf{A}] \mathbf{x}_k - \alpha \mathbf{d}$$

Устойчивость процесса поиска минимума определяется
собственными значениями λ_i матрицы $[\mathbf{I} - \alpha \mathbf{A}]$

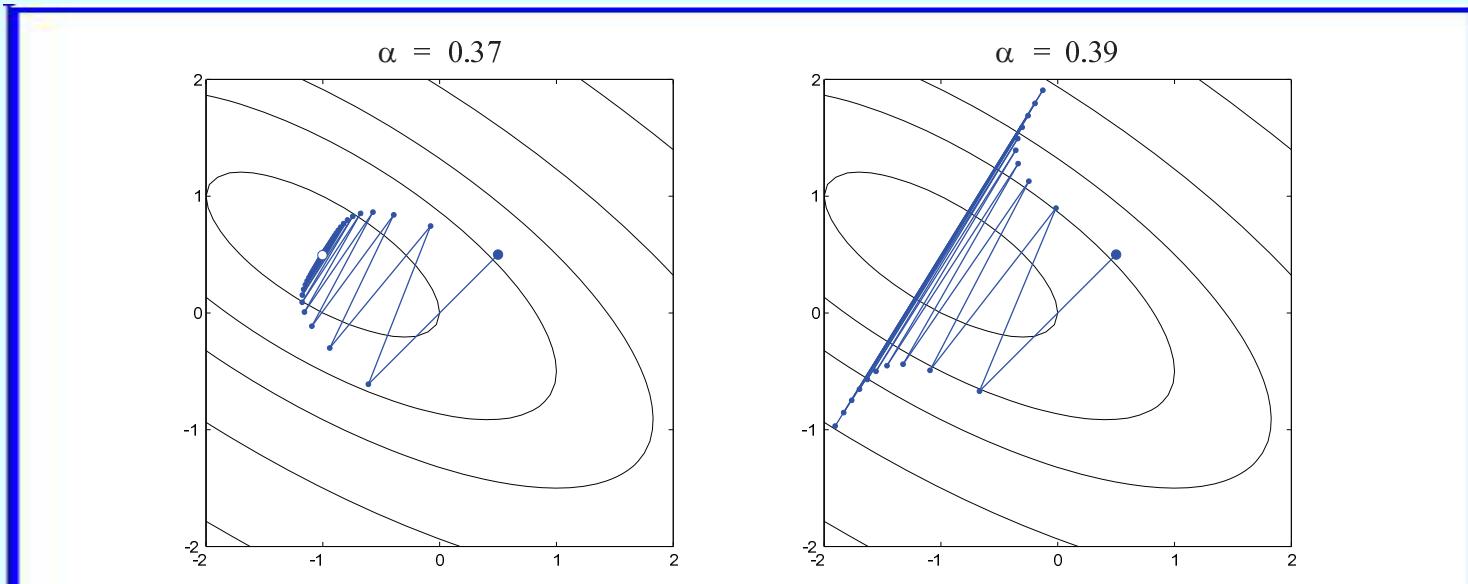
Условие устойчивости:

$$|(1 - \alpha \lambda_i)| < 1, \quad \alpha < \frac{2}{\lambda_i}, \quad \alpha < \frac{2}{\lambda_{max}}$$

Применение сетей прямого распространения (XL)

Недостатки исходного алгоритма ВР – 9

Переход к переменной скорости обучения (3)



Алгоритм SDBP с постоянной скоростью обучения

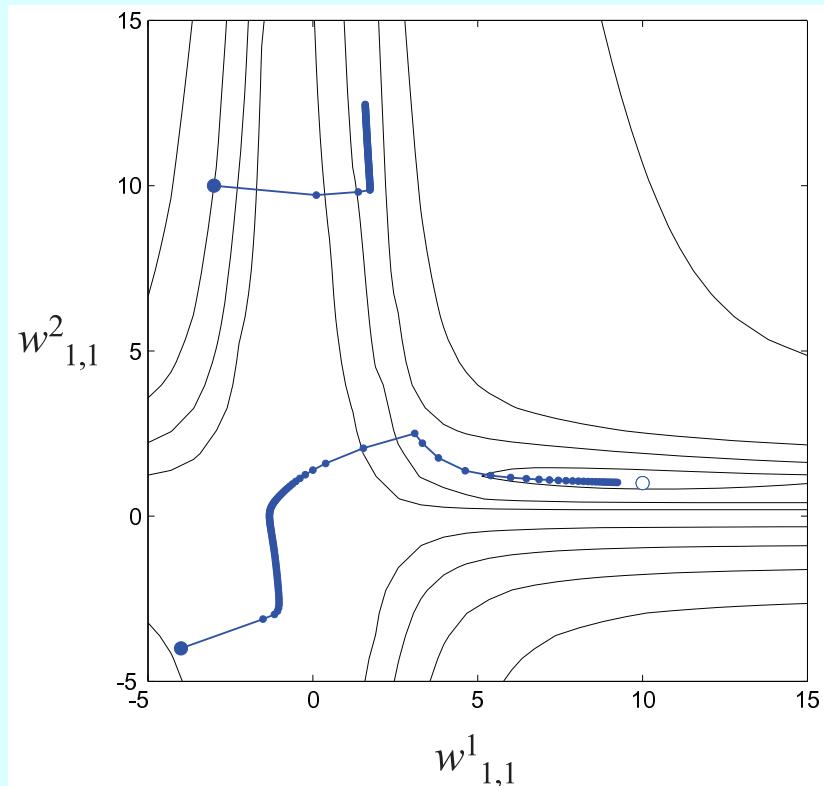
$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$$

$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}, \quad \lambda_1 = 0.764, \quad \lambda_2 = 5.24; \quad \alpha < \frac{2}{\lambda_{max}} = \frac{2}{5.24} = 0.38$$

Применение сетей прямого распространения (XLI)

Недостатки исходного алгоритма ВР – 10

Переход к переменной скорости обучения (4)



Сходимость алгоритма SDBP
при $\alpha = \text{const}$

Нижняя траектория:

Переход из **первой** стартовой точки в направлении точки **(10, 1)** – **глобальный минимум** функции ошибки $E(w_1^1, w_2^1)$.

Верхняя траектория:

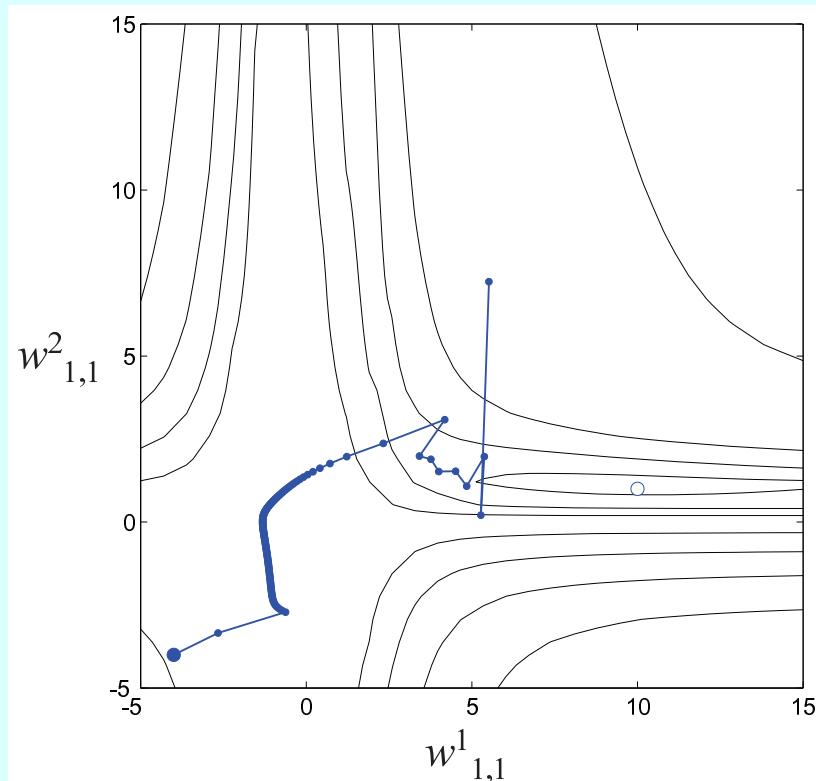
Переход из **второй** стартовой точки в направлении точки **(0.88, 38.6)** – **локальный минимум** функции ошибки $E(w_1^1, w_2^1)$ (переход не завершен).

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figure 12.6, p.12-8).

Применение сетей прямого распространения (XLII)

Недостатки исходного алгоритма ВР – 11

Переход к переменной скорости обучения (5)



Сходимость алгоритма SDBP
при $\alpha = \text{const}$

Переход из стартовой точки в направлении точки **(10, 1)** — **глобальный минимум** функции ошибки $E(w_{1,1}^1, w_{1,1}^2)$.

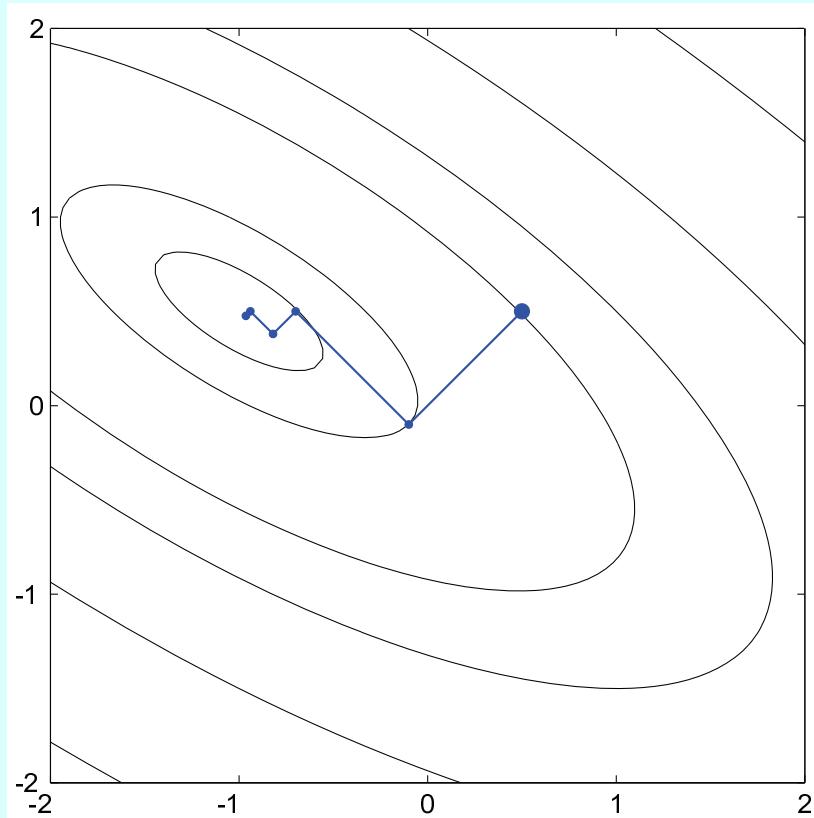
Превышена допустимая скорость обучения
 α — алгоритм потерял устойчивость.

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figure 12.8, p.12-9).

Применение сетей прямого распространения (XLIII)

Недостатки исходного алгоритма ВР – 12

Переход к переменной скорости обучения (6)



Оптимизация одномерного поиска по α

Задача оптимизации:

- формирование направления поиска,
- одномерная оптимизация вдоль направления поиска.

При использовании одномерной оптимизации на каждом шаге — направления поиска ортогональны.

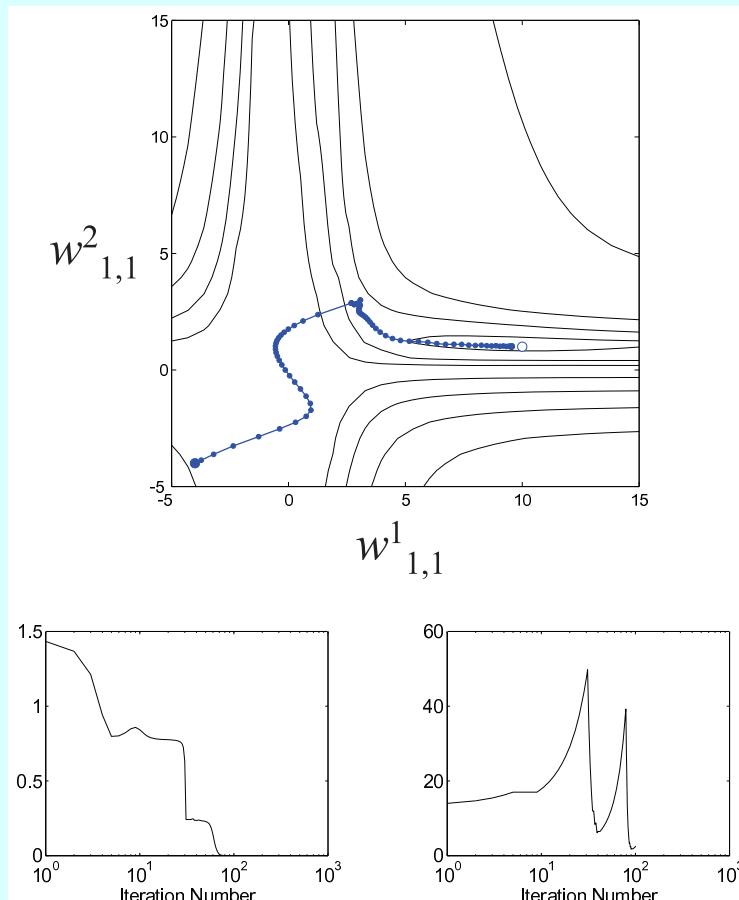
Метод обладает высокой эффективностью при квадратичной минимизируемой функции.

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.4, p.9-10).

Применение сетей прямого распространения (XLIV)

Недостатки исходного алгоритма ВР – 13

Переход к переменной скорости обучения (7)



Траектория поиска решения
алгоритмом SDBP при $\alpha = \text{var}$

Если при выполнении последовательности шагов поиска минимума темп уменьшения функции ошибки $E(x)$ снижается, следует увеличить скорость обучения α .

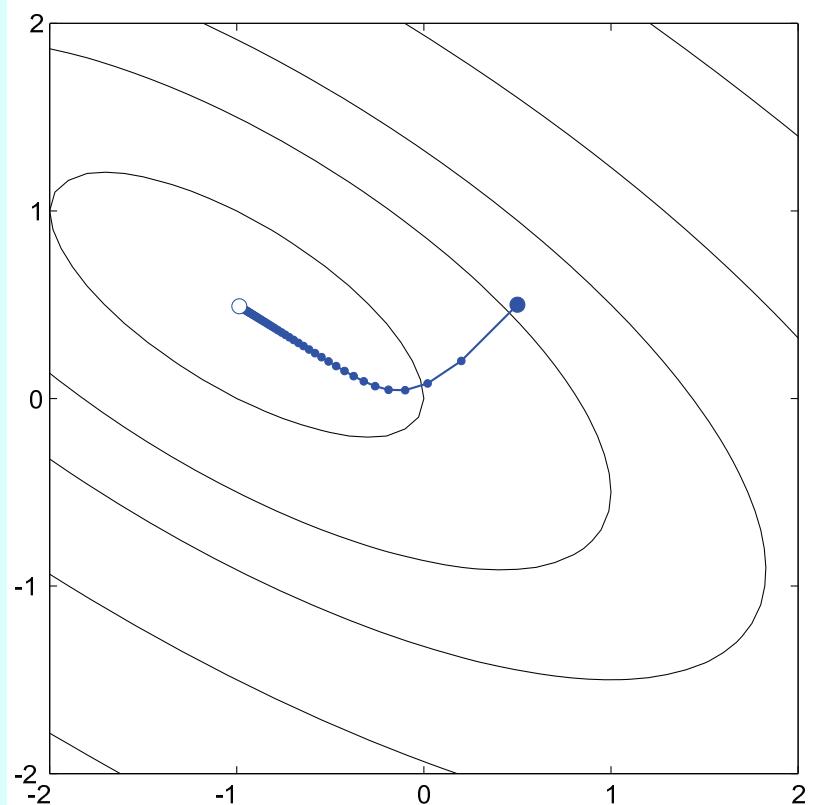
Если функция ошибки $E(x)$ растет, следует отменить результаты выполнения шага, уменьшить скорость обучения α и исполнить шаг повторно.

Изменение по числу выполненных итераций
квадратичной ошибки сети (слева) и
скорости обучения (справа).

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figures 12.11 and 12.12, p.12-13).

Применение сетей прямого распространения (ХЛВ)

Недостатки исходного алгоритма ВР – 14



Общая схема итерации, реализуемой алгоритмом ВР:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k$$

Здесь \mathbf{p}_k – направление поиска минимума функции ошибки сети.

Направление поиска, реализуемое SDBP:

$$\mathbf{p}_k = -\mathbf{g}_k,$$

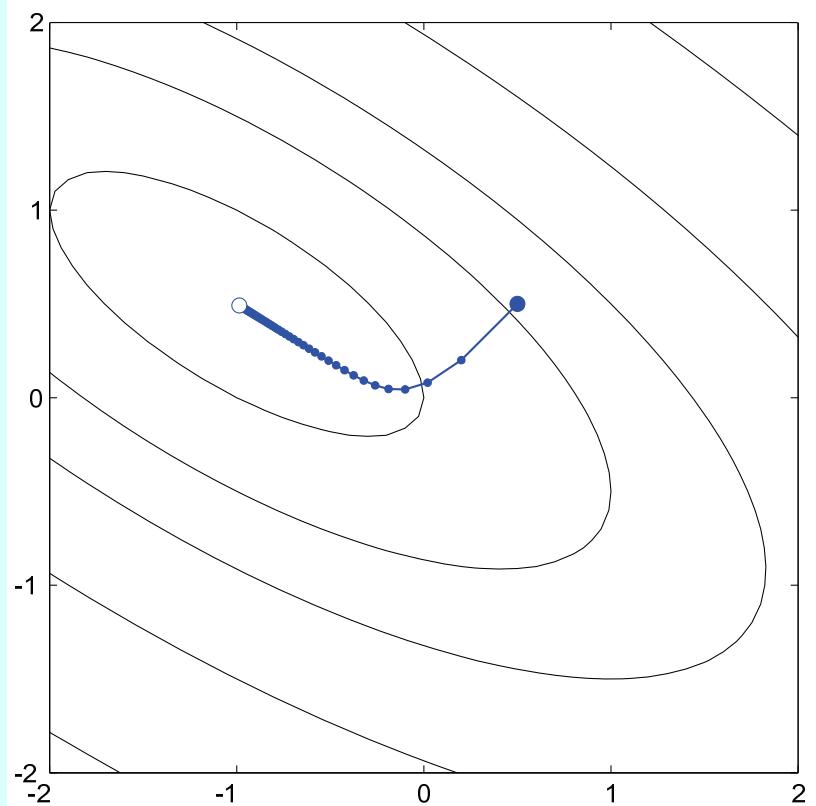
$$\mathbf{g}_k = \nabla E(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_k}$$

\mathbf{g}_k – градиент функции ошибки $E(\mathbf{w})$
в точке \mathbf{w}_k

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.10, p.9-20).

Применение сетей прямого распространения (XLVI)

Недостатки исходного алгоритма ВР – 15



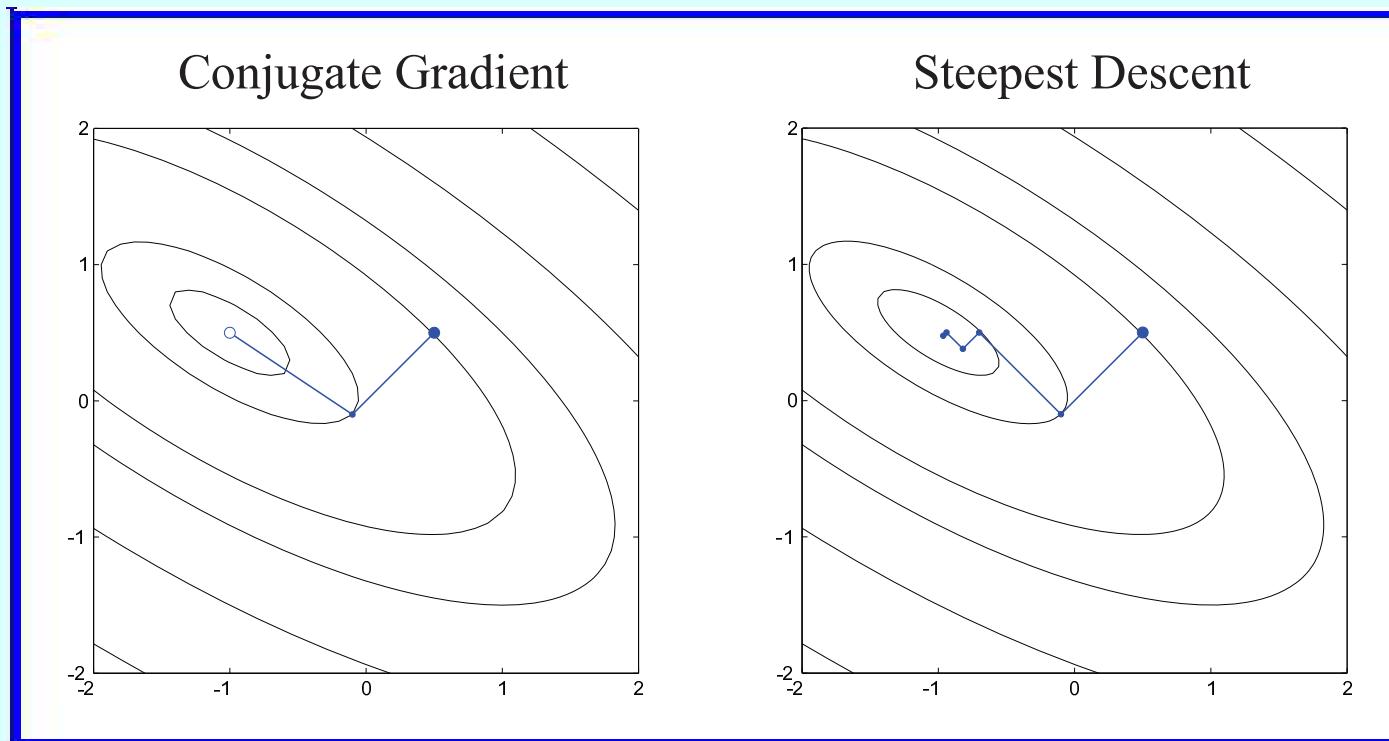
Использование оптимизационных схем, отличных от SDBP:

- алгоритм сопряженных градиентов в различных вариантах (CGBP — Conjugate Gradient Back-Propagation) для корректировки направления поиска минимума функции ошибки, в сравнении с задаваемым алгоритмом SDBP;
- алгоритм Левенберга-Марквардта (LMBP — Levenberg-Marquardt Back-Propagation), позволяющий учитывать не только направление убывания функции ошибки, но и кривизну поверхности, задаваемой функцией ошибки, в текущей точке процесса поиска решения.

Источник: Hagan M. T., Demuth H.B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.1, p.9-5).

Применение сетей прямого распространения (XLVII)

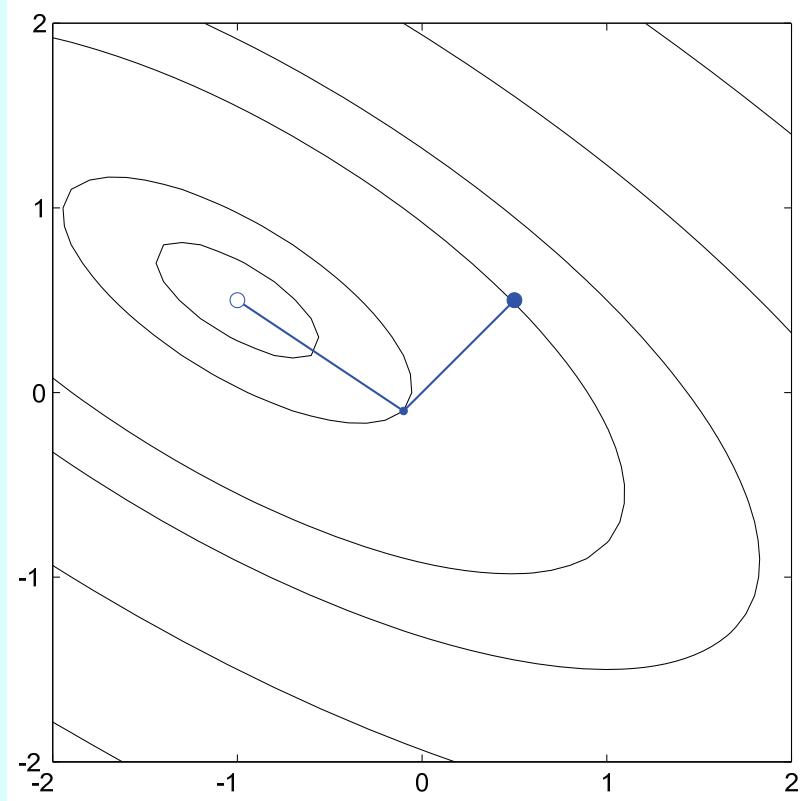
Недостатки исходного алгоритма ВР – 16



Источник: *Hagan M. T., Demuth H.B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figures 9.1 and 9.10, pp.9-5, 9-20).

Применение сетей прямого распространения (XLVIII)

Недостатки исходного алгоритма ВР – 17



Алгоритм CGBP

Общая схема итерации CGBP:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k$$

Направление поиска,
реализуемое CGBP:

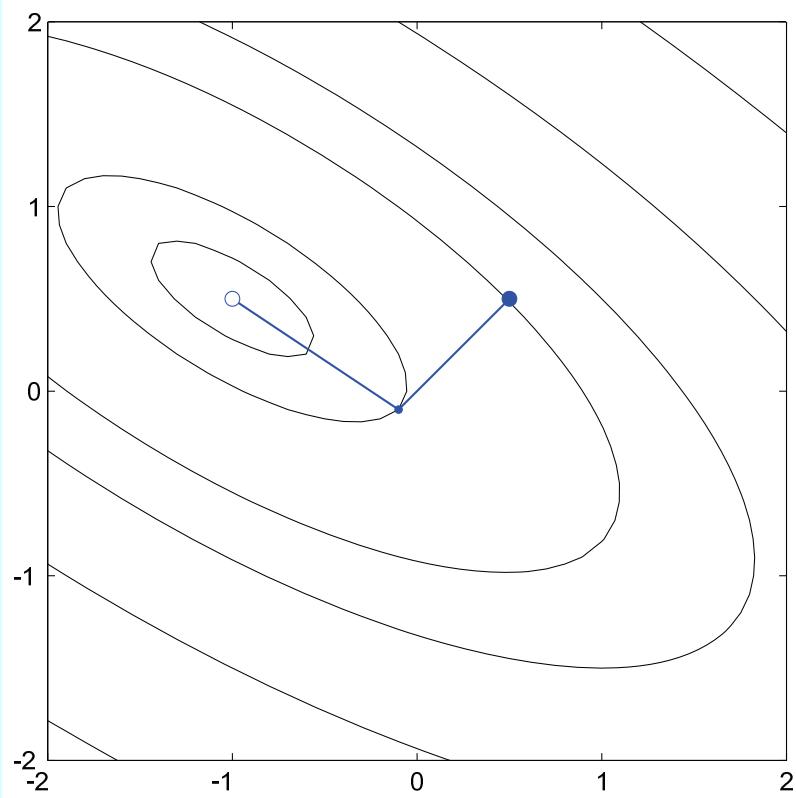
$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1},$$

$$\mathbf{g}_k = \nabla E(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_k}$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.10, p.9-20).

Применение сетей прямого распространения (XLIX)

Недостатки исходного алгоритма ВР – 18



Варианты алгоритма CGBP

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k$$

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1},$$

$$\mathbf{g}_k = \nabla E(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_k}$$

Hestenes-Steifel:

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\Delta \mathbf{g}_{k-1}^T \mathbf{p}_{k-1}}$$

Fletcher-Reeves:

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$

Polak-Ribière:

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.10, p.9-20).

Применение сетей прямого распространения (L)

Недостатки исходного алгоритма ВР – 19

Метод Ньютона (1)

Оптимизационные алгоритмы для функции $F(\mathbf{x}) = F(x_1, x_2, \dots, x_n)$ основаны на разложении этой функции в ряд Тейлора в окрестности точки \mathbf{x}^* :

$$\begin{aligned} F(\mathbf{x}) &= F(\mathbf{x}^*) + \frac{\partial}{\partial x_1} F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*) + \frac{\partial}{\partial x_2} F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} (x_2 - x_2^*) + \\ &+ \dots + \frac{\partial}{\partial x_n} F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} (x_n - x_n^*) + \frac{1}{2} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*)^2 + \\ &+ \dots + \frac{1}{2} \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*)(x_2 - x_2^*) + \dots \end{aligned}$$

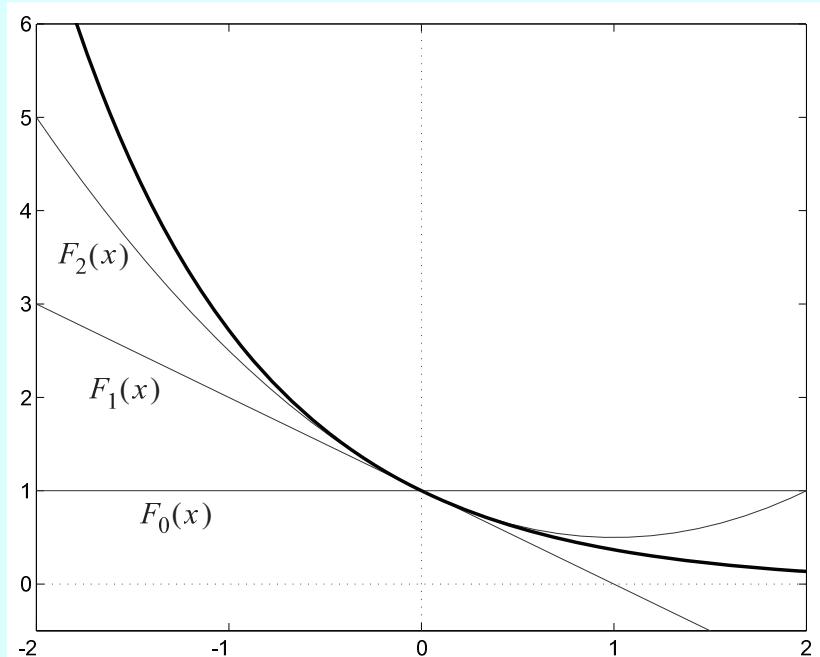
В матричной форме:

$$\begin{aligned} F(\mathbf{x}) &= F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \\ &+ \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^T \nabla^2 F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \dots \end{aligned}$$

Применение сетей прямого распространения (LI)

Недостатки исходного алгоритма ВР – 20

Метод Ньютона (2)



Пример аппроксимации рядом Тейлора

$$F(x) = e^{-x}, \quad x^* = 0$$

$$F(x) = e^{-x} = e^{-0} - e^{-0}(x - 0) +$$

$$+ \frac{1}{2}e^{-0}(x - 0)^2 + \frac{1}{6}e^{-0}(x - 0)^3 + \dots$$

$$F(x) = 1 - x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots$$

$$F(x) \approx F_0(x) = 1$$

$$F(x) \approx F_1(x) = 1 - x$$

$$F(x) \approx F_2(x) = 1 - x + \frac{1}{2}x^2$$

Применение сетей прямого распространения (LII)

Недостатки исходного алгоритма ВР – 21

Метод Ньютона (3)

В разложении в ряд Тейлора $\nabla F(\mathbf{x})$ — градиент:

$$\nabla F(\mathbf{x}) = \left[\frac{\partial}{\partial x_1} F(\mathbf{x}) \quad \frac{\partial}{\partial x_2} F(\mathbf{x}) \quad \dots \quad \frac{\partial}{\partial x_n} F(\mathbf{x}) \right]^T$$

В этом разложении $\nabla^2 F(\mathbf{x})$ — матрица Гессе (гессиан):

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_1 \partial x_n} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_2 \partial x_n} F(\mathbf{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_n^2} F(\mathbf{x}) \end{bmatrix}$$

Применение сетей прямого распространения (LIII)

Недостатки исходного алгоритма ВР – 22

Метод Ньютона (4)

Алгоритм наискорейшего спуска основан на использовании линейной аппроксимации разложения минимизируемой функции $F(\mathbf{x})$ в ряд Тейлора:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx \mathbf{F}(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k$$

При использовании квадратичной аппроксимации получим метод Ньютона:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx \mathbf{F}(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k + \frac{1}{2} \Delta\mathbf{x}_k^T \mathbf{A} \Delta\mathbf{x}_k$$

Применение сетей прямого распространения (LIV)

Недостатки исходного алгоритма ВР – 23

Метод Ньютона (5)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k$$

$$\mathbf{A}_k \equiv \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \quad \mathbf{g}_k \equiv \nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k}$$

Для квадратичного критерия

$$F(\mathbf{x}) = \sum_{i=1}^N v_i^2(\mathbf{x}) = \mathbf{v}^T(\mathbf{x}) \mathbf{v}(\mathbf{x})$$

j-й элемент градиента:

$$[\nabla F(\mathbf{x})]_j = \frac{\partial F(\mathbf{x})}{\partial x_j} = 2 \sum_{i=1}^N v_i(\mathbf{x}) \frac{\partial v_i(\mathbf{x})}{\partial x_j}$$

Применение сетей прямого распространения (LV)

Недостатки исходного алгоритма ВР – 24

Метод Ньютона (6)

Градиент в матричной форме:

$$\nabla F(\mathbf{x}) = \mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x})$$

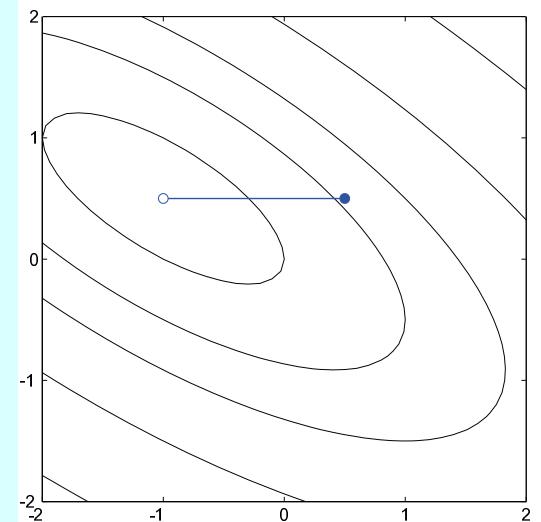
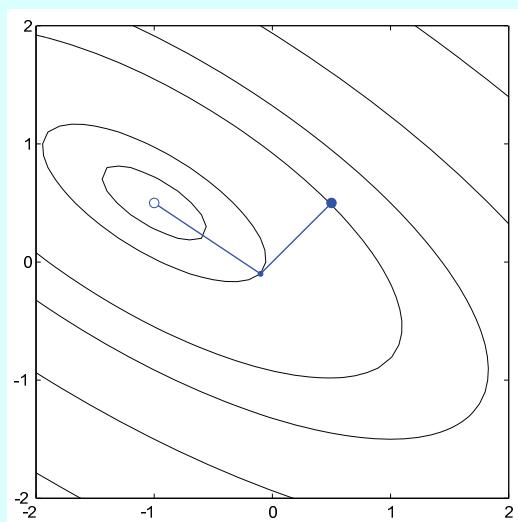
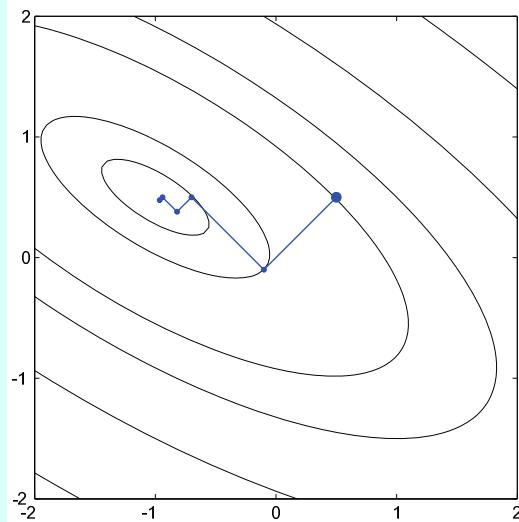
Здесь **J** матрица Якоби (якобиан):

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1(\mathbf{x})}{\partial x_1} & \frac{\partial v_1(\mathbf{x})}{\partial x_2} & \dots & \frac{\partial v_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial v_2(\mathbf{x})}{\partial x_1} & \frac{\partial v_2(\mathbf{x})}{\partial x_2} & \dots & \frac{\partial v_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial v_N(\mathbf{x})}{\partial x_1} & \frac{\partial v_N(\mathbf{x})}{\partial x_2} & \dots & \frac{\partial v_N(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Применение сетей прямого распространения (LVI)

Недостатки исходного алгоритма ВР – 25

Метод Ньютона (7)



Характер процесса поиска решения методами **наискорейшего спуска** (слева),
сопряженных градиентов (в центре) и **Ньютона** (справа).

Источник: *Hagan M.T., Demuth H.B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 9, Figure 9.4, p.9-10; Figure 9.10, p.9-20; Figure 9.5, p.9-12).

Применение сетей прямого распространения (LVII)

Недостатки исходного алгоритма ВР – 26

Метод Ньютона (8)

Матрица Гессе, выраженная через матрицу Якоби:

$$[\nabla^2 F(\mathbf{x})]_{k,j} = \frac{\partial^2 F(\mathbf{x})}{\partial x_k \partial x_j} = 2 \sum_{i=1}^N \left\{ \frac{\partial v_i(\mathbf{x})}{\partial x_k} \cdot \frac{\partial v_i(\mathbf{x})}{\partial x_j} + v_i(\mathbf{x}) \frac{\partial^2 v_i(\mathbf{x})}{\partial x_k \partial x_j} \right\}$$

$$\nabla^2 F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\mathbf{S}(\mathbf{x})$$

$$\mathbf{S}(\mathbf{x}) = \sum_{i=1}^N v_i(\mathbf{x}) \nabla^2 v_i(\mathbf{x})$$

Приближенное выражение для матрицы Гессе
в предположении, что $\mathbf{S}(\mathbf{x})$ мало:

$$\nabla^2 F(\mathbf{x}) \approx 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x})$$

Применение сетей прямого распространения (LVIII)

Недостатки исходного алгоритма ВР – 27

Переход к методу Гаусса-Ньютона

Метод Ньютона:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k$$

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \quad \mathbf{A}_k \equiv \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k}$$



$$\nabla F(\mathbf{x}) = \mathbf{J}^T(\mathbf{x}) \mathbf{v}(\mathbf{x}) \quad \nabla^2 F(\mathbf{x}) \approx 2\mathbf{J}^T(\mathbf{x}) \mathbf{J}(\mathbf{x})$$

Метод Гаусса-Ньютона:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [2\mathbf{J}^T(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k)]^{-1} 2\mathbf{J}^T(\mathbf{x}_k) \mathbf{v}(\mathbf{x}_k) =$$

$$= \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k)]^{-1} \mathbf{J}^T(\mathbf{x}_k) \mathbf{v}(\mathbf{x}_k)$$

В методе Гаусса-Ньютона, в отличие от метода Ньютона, **не требуется вычислять вторые производные** минимизируемой функции.

Применение сетей прямого распространения (LIX)

Недостатки исходного алгоритма ВР – 28

Переход к методу Левенберга-Марквардта

Аппроксимация матрицы Гессе H ,
используемая в методе Гаусса-Ньютона:

$$H = J^T J$$

Матрица H может оказаться **вырожденной**.

Модификация матрицы H , чтобы она стала **обращаемой**:

$$G = H + \mu I, \quad I \text{ — единичная матрица}$$

Метод Левенберга-Марквардта:

$$x_{k+1} = x_k - [J^T(x_k)J(x_k) + \mu_k I]^{-1} J^T(x_k) v(x_k)$$

Применение сетей прямого распространения (ЛХ)

Недостатки исходного алгоритма ВР – 29

Метод Левенберга-Марквардта (LM):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k \mathbf{I}]^{-1} \mathbf{J}^T(\mathbf{x}_k) \mathbf{v}(\mathbf{x}_k)$$

Если $\mu_k \rightarrow 0$, метод Левенберга-Марквардта превращается в **метод Гаусса-Ньютона (GN)**:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1} \mathbf{J}^T(\mathbf{x}_k) \mathbf{v}(\mathbf{x}_k)$$

Если $\mu_k \rightarrow \infty$, метод Левенберга-Марквардта превращается в **метод наискорейшего спуска (SD)**:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{\mu_k} \mathbf{J}^T(\mathbf{x}_k) \mathbf{v}(\mathbf{x}_k) = \mathbf{x}_k - \frac{1}{\mu_k} \nabla F(\mathbf{x})$$

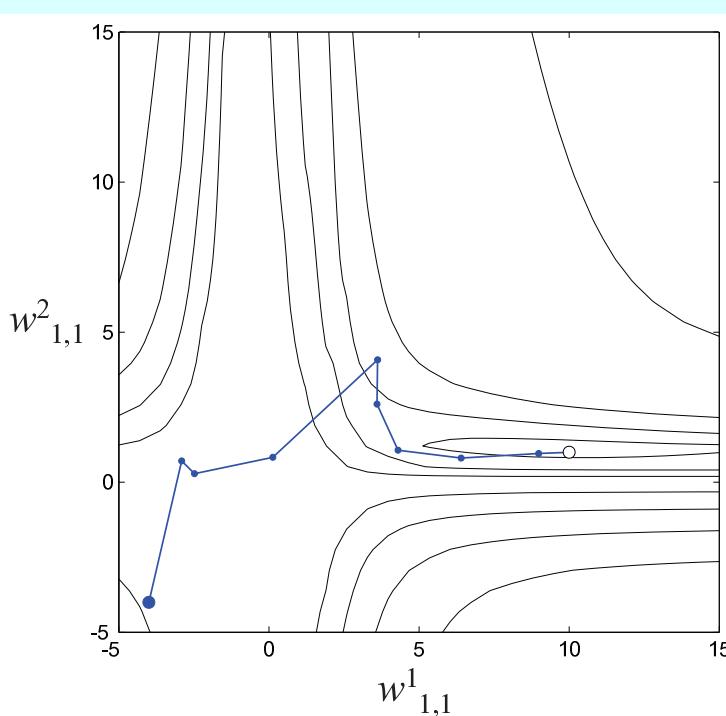
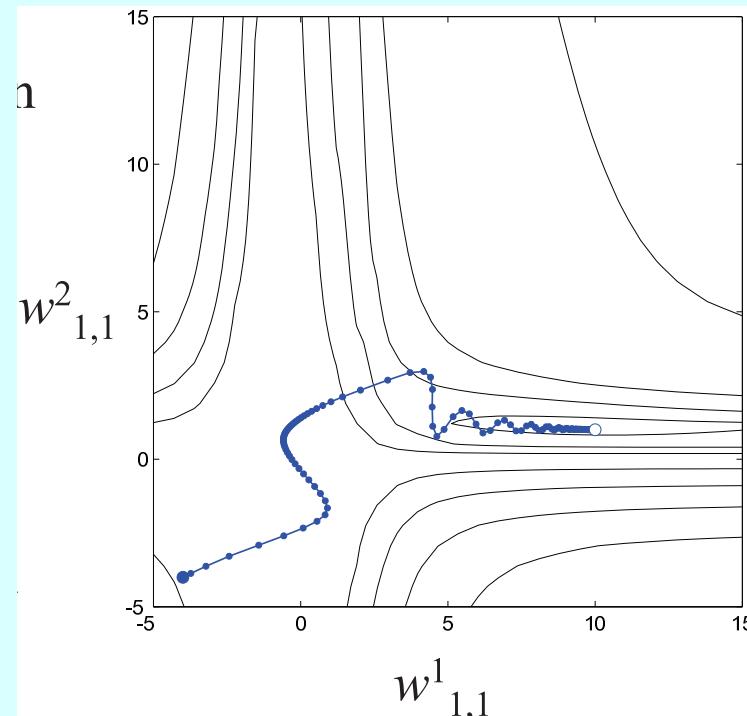
Варьирование μ_k :

- при **малых** μ_k метод LM ведет себя аналогично методу GN;
- при **больших** μ_k метод LM ведет себя аналогично методу SD.

Применение сетей прямого распространения (LXI)

Недостатки исходного алгоритма ВР – 30

Сравнение методов наискорейшего спуска и Левенберга-Марквардта



Траектория поиска решения,
реализуемая методом наискорейшего спуска с моментом (слева)
и методом Левенберга-Марквардта (справа).

Применение сетей прямого распространения (LXII)

Недостатки исходного алгоритма ВР – 31

Метод Левенберга-Марквардта в обучении сетей (1)

Функция ошибки для многослойной сети:

$$E(x) = \sum_{q=1}^Q (t_q - a_q)^T (t_q - a_q) = \sum_{q=1}^Q e_q^T e_q = \sum_{q=1}^Q \sum_{j=1}^{S^M} (e_{j,q})^2 = \sum_i^N (v_i)^2$$

Вектор ошибок:

$$v^T = [v_1 \ v_2 \ \dots \ v_N] = [e_{1,1} \ e_{2,1} \ \dots \ e_{S^M,1} \ e_{1,2} \ \dots \ e_{S^M,Q}]$$

Вектор параметров:

$$x^T = [x_1 \ x_2 \ \dots \ x_n] = [w_{1,1}^1 \ w_{1,2}^1 \ \dots \ w_{S^1,R}^1 \ b_1^1 \ \dots \ b_{S^1}^1 \ w_{1,1}^2 \ \dots \ b_{S^M}^M]$$

Размерности векторов v и x :

$$N = Q \times S^M, \quad n = S^1(R+1) + S^2(S^1+1) + \dots + S^M(S^{M-1}+1)$$

Применение сетей прямого распространения (LXIII)

Недостатки исходного алгоритма ВР – 32

Метод Левенберга-Марквардта в обучении сетей (2)

Функция ошибки для многослойной сети:

$$E(x) = \sum_{q=1}^Q (t_q - a_q)^T (t_q - a_q) = \sum_{q=1}^Q e_q^T e_q = \sum_{q=1}^Q \sum_{j=1}^{S^M} (e_{j,q})^2 = \sum_i^N (v_i)^2$$

Матрица Якоби:

$$J(x) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_{1,1}^1} & \frac{\partial e_{1,1}}{\partial w_{1,2}^1} & \dots & \frac{\partial e_{1,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,1}}{\partial b_1^1} & \dots \\ \frac{\partial e_{2,1}}{\partial w_{1,1}^1} & \frac{\partial e_{2,1}}{\partial w_{1,2}^1} & \dots & \frac{\partial e_{2,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{2,1}}{\partial b_1^1} & \dots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{S^M,1}}{\partial w_{1,1}^1} & \frac{\partial e_{S^M,1}}{\partial w_{1,2}^1} & \dots & \frac{\partial e_{S^M,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{S^M,1}}{\partial b_1^1} & \dots \\ \frac{\partial e_{1,2}}{\partial w_{1,1}^1} & \frac{\partial e_{1,2}}{\partial w_{1,2}^1} & \dots & \frac{\partial e_{1,2}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,2}}{\partial b_1^1} & \dots \\ \vdots & \vdots & & \vdots & \vdots & \end{bmatrix}$$

Применение сетей прямого распространения (LXIV)

Недостатки исходного алгоритма ВР – 33

Метод Левенберга-Марквардта в обучении сетей (3)

Элементы матрицы Якоби, требуемые для использования метода Левенберга-Марквардта, можно вычислить с помощью модифицированного алгоритма обратного распространения ошибки.

Стандартный алгоритм обратного распространения (**SDBP**) вычисляет частные производные функции ошибки по настраиваемым параметрам сети

$$\frac{\partial E(x)}{\partial x_i} = \frac{\partial e_q^T e_q}{\partial x_i}$$

используя цепное правило

$$\frac{\partial E}{\partial w_{i,j}^m} = \frac{\partial E}{\partial n_j^m} \times \frac{\partial n_j^m}{\partial w_{i,j}^m}$$

Невязки (чувствительности) в алгоритме **SDBP**

$$\delta_i^m \equiv \frac{\partial E}{\partial n_j^m}$$

вычисляются в процессе обратного распространения ошибки сети от ее выходов к входам.

Применение сетей прямого распространения (LXV)

Недостатки исходного алгоритма ВР – 34

Метод Левенберга-Марквардта в обучении сетей (4)

Элементы матрицы Якоби:

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial x_l}$$

Невязка Марквардта для вычисления элементов матрицы Якоби:

$$\tilde{\delta}_{i,h}^m \equiv \frac{\partial v_h}{\partial n_{i,q}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m}, \quad h = (q-1)S^M + k$$

Вычисление элементов матрицы Якоби через невязки Марквардта:

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial w_{i,j}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = \tilde{\delta}_{i,h}^m \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = \tilde{\delta}_{i,h}^m \times a_{j,q}^{m-1}$$

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial b_i^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = \tilde{\delta}_{i,h}^m \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = \tilde{\delta}_{i,h}^m$$

Применение сетей прямого распространения (LXVI)

Недостатки исходного алгоритма ВР – 35

Метод Левенберга-Марквардта в обучении сетей (5)

Невязка выходного слоя в алгоритме SDBP:

$$\delta_i^M = -(t_i - a_i^M)a_i^M(1 - a_i^M)$$

Невязка Марквардта для вычисления элементов матрицы Якоби:

$$\tilde{\delta}_{i,h}^m \equiv \frac{\partial v_h}{\partial n_{i,q}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m}, \quad h = (q-1)S^M + k$$

Невязка Марквардта выходного слоя в алгоритме LMBP:

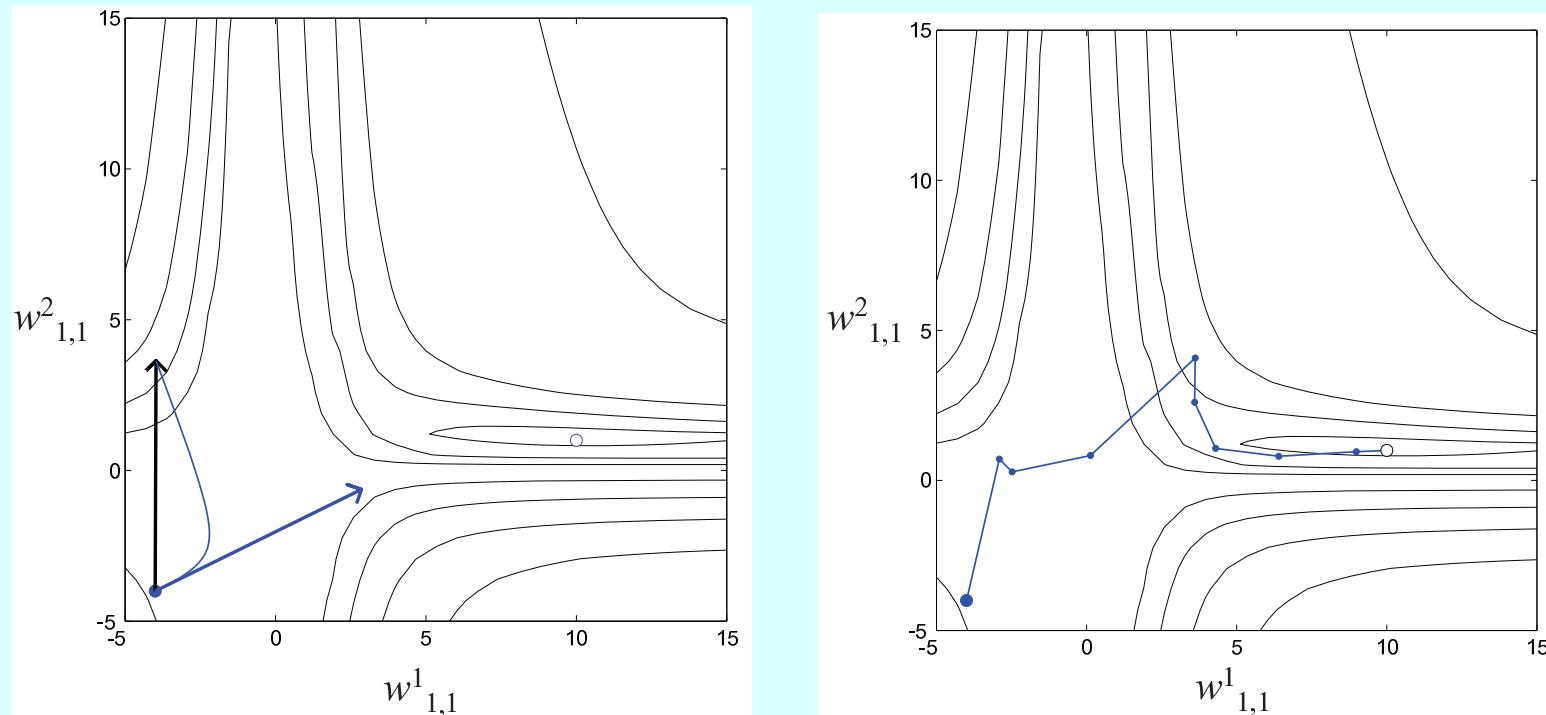
$$\tilde{\delta}_{i,h}^M \equiv \frac{\partial v_h}{\partial n_{i,q}^M} = \frac{\partial e_{k,q}}{\partial n_{i,q}^M} = \frac{\partial(t_{k,q} - a_{k,q}^M)}{\partial n_{i,q}^M} = -\frac{\partial a_{k,q}^M}{\partial n_{i,q}^M}$$

Невязка в алгоритме LMBP вычисляется в целом аналогично тому, как это делается в алгоритме SDBP, но частные производные берутся для ошибок $e_{i,q}$ (это j -й элемент ошибки для q -го примера).

Применение сетей прямого распространения (LXVII)

Недостатки исходного алгоритма ВР – 36

Работа метода Левенберга-Марквардта



Слева: один шаг метода LM: **черная стрелка** — направление поиска при малых μ (соответствует методу Гаусса-Ньютона); **синяя стрелка** — направление поиска при больших μ (соответствует наискорейшему спуску).

Справа: работа метода LM при стартовом $\mu_0 = 0.01$.

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 12, Figures 12.17 and 12.18, pp.12-26, 12-27).

Применение сетей прямого распространения (LXVIII)

Тестирование методов обучения – 1

Методы обучения

Acronym	Algorithm	
LM	trainlm	Levenberg-Marquardt
BFG	trainbfg	BFGS Quasi-Newton
RP	trainrp	Resilient Backpropagation
SCG	trainscg	Scaled Conjugate Gradient
CGB	traincgb	Conjugate Gradient with Powell/Beale Restarts
CGF	traincfg	Fletcher-Powell Conjugate Gradient
CGP	traincgp	Polak-Ribiére Conjugate Gradient
OSS	trainoss	One Step Secant
GDX	traingdx	Variable Learning Rate Backpropagation

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-34).

Применение сетей прямого распространения (LXIX)

Тестирование методов обучения – 2

Пример аппроксимации функций (1)

Algorithm	Mean Time (s)	Ratio	Min. Time (s)	Max. Time (s)	Std. (s)
LM	18.45	1.00	12.01	30.03	4.27
BFG	27.12	1.47	16.42	47.36	5.95
SCG	36.02	1.95	19.39	52.45	7.78
CGF	37.93	2.06	18.89	50.34	6.12
CGB	39.93	2.16	23.33	55.42	7.50
CGP	44.30	2.40	24.99	71.55	9.89
OSS	48.71	2.64	23.51	80.90	12.33
RP	65.91	3.57	31.83	134.31	34.24
GDX	188.50	10.22	81.59	279.90	66.67

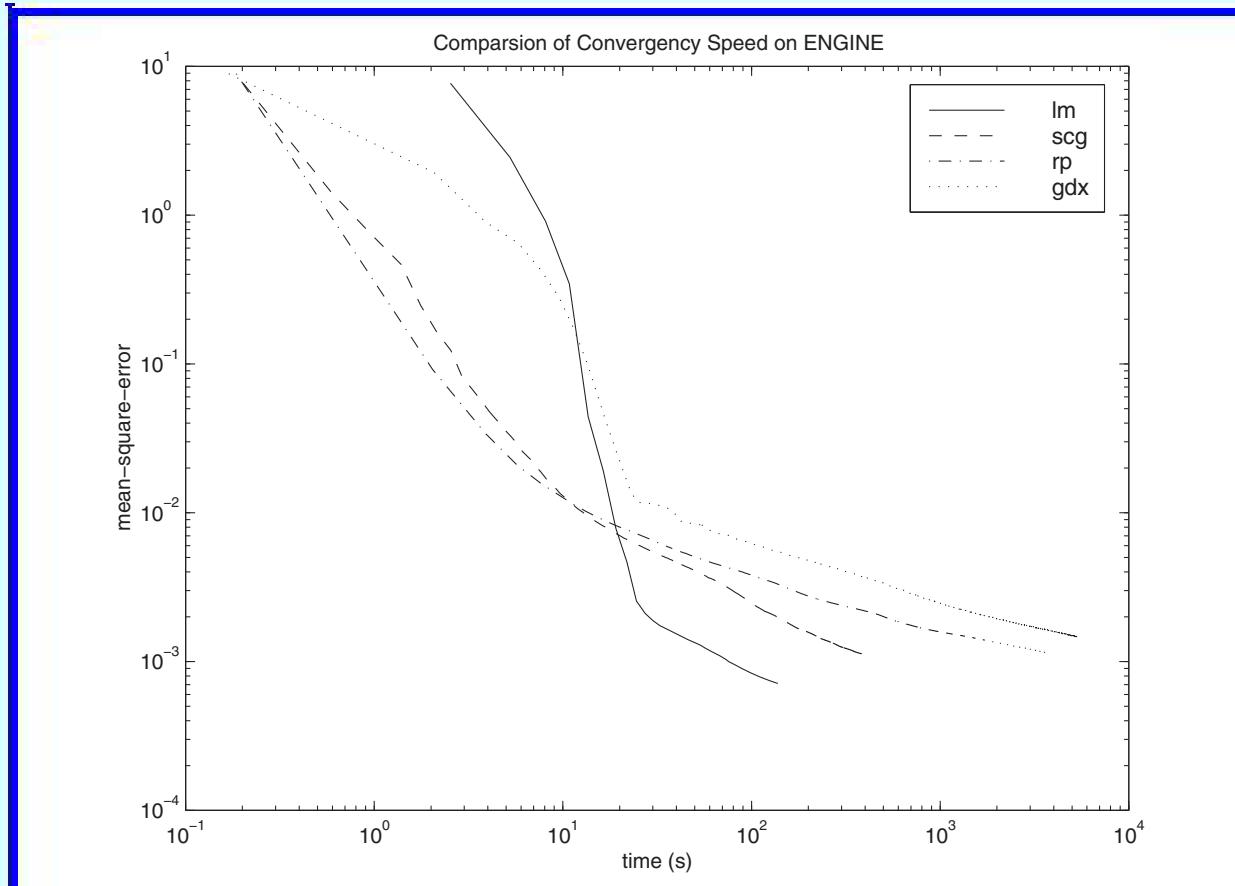
Нелинейная регрессия, сеть со структурой **2-30-2**, сигмоидальные нейроны (гиперболический тангенс) в скрытом слое, линейные нейроны в выходном слое.

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-40).

Применение сетей прямого распространения (LXX)

Тестирование методов обучения – 3

Пример аппроксимации функций (2)

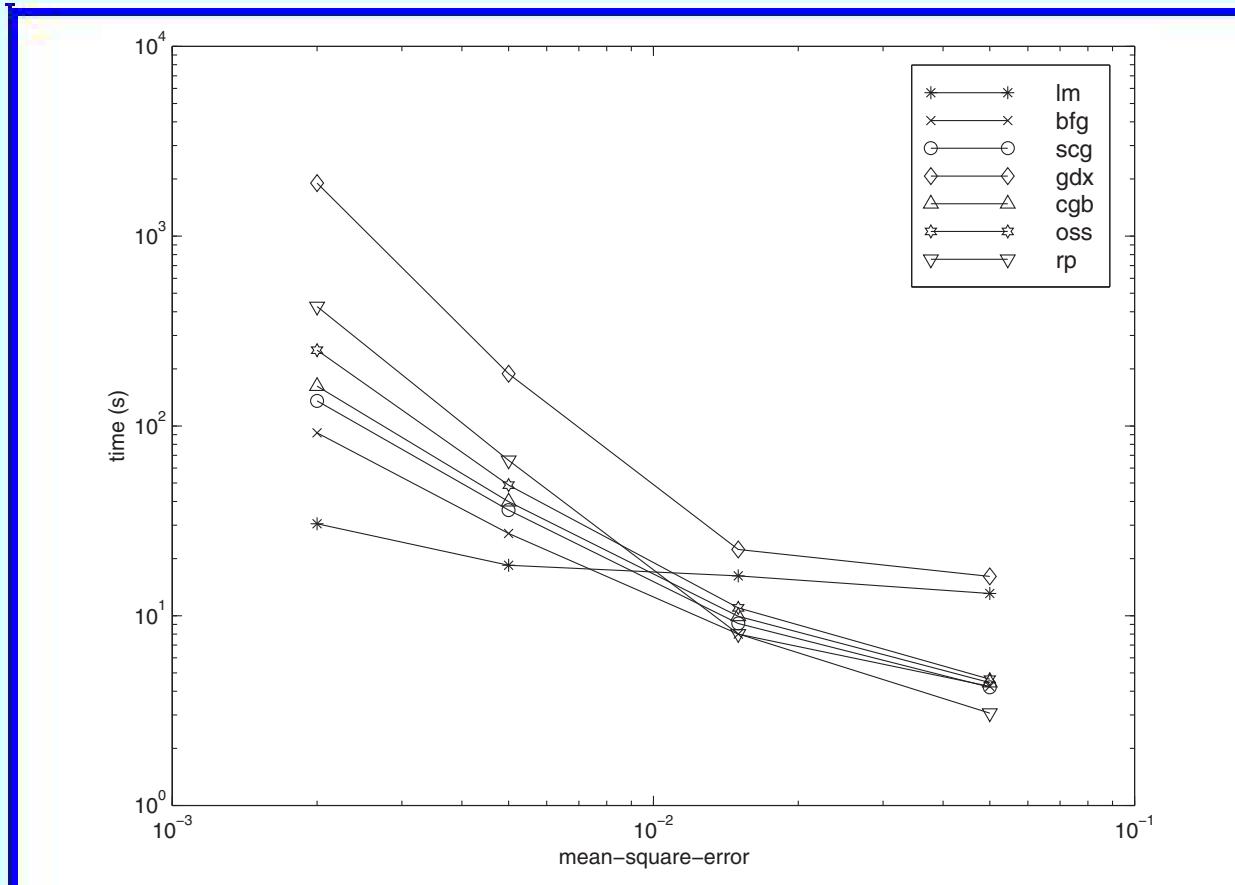


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-41).

Применение сетей прямого распространения (LXXI)

Тестирование методов обучения – 4

Пример аппроксимации функций (3)



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-42).

Применение сетей прямого распространения (LXXII)

Тестирование методов обучения – 5

Пример задачи классификации (1)

Algorithm	Mean Time (s)	Ratio	Min. Time (s)	Max. Time (s)	Std. (s)
RP	3.73	1.00	2.35	6.89	1.26
SCG	4.09	1.10	2.36	7.48	1.56
CGP	5.13	1.38	3.50	8.73	1.05
CGB	5.30	1.42	3.91	11.59	1.35
CGF	6.62	1.77	3.96	28.05	4.32
OSS	8.00	2.14	5.06	14.41	1.92
LM	13.07	3.50	6.48	23.78	4.96
BFG	19.68	5.28	14.19	26.64	2.85
GDX	27.07	7.26	25.21	28.52	0.86

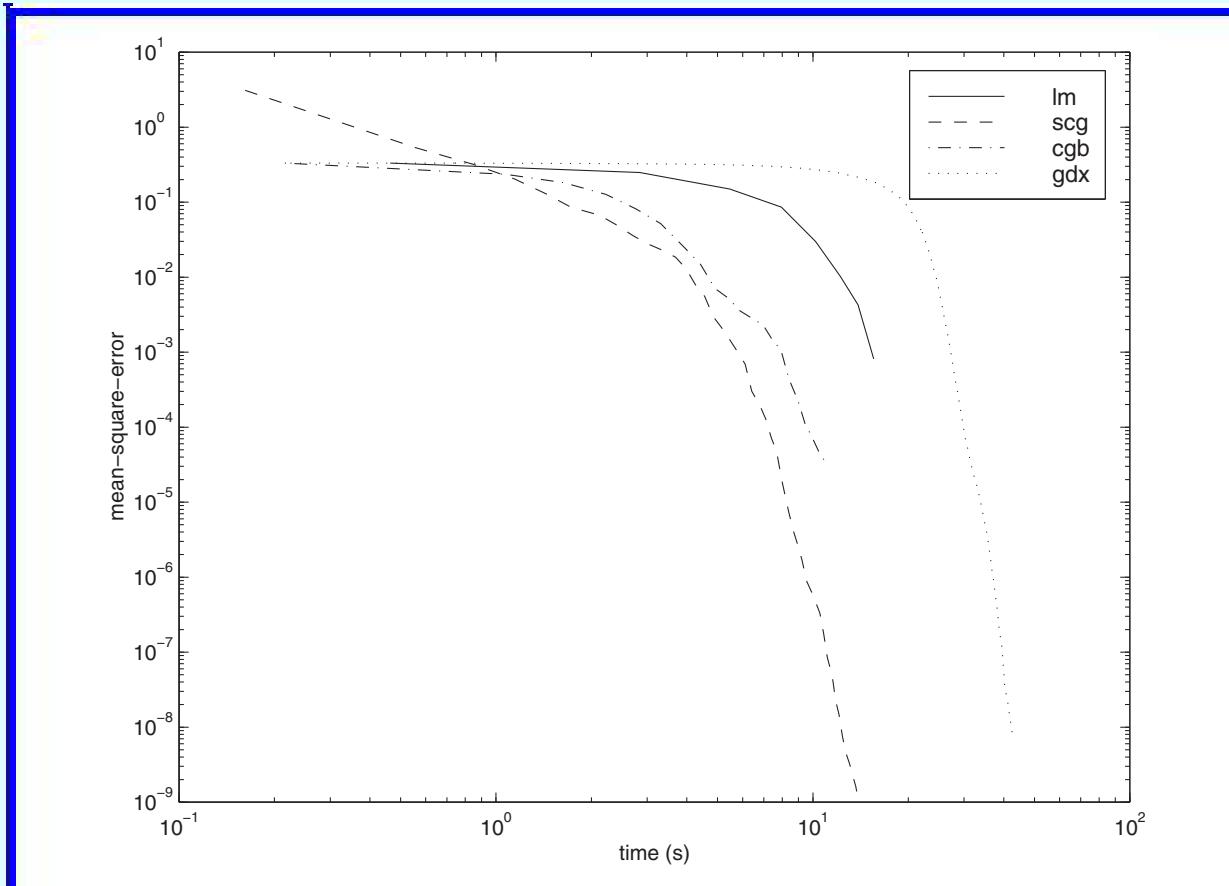
Разделение чисел длиною три двоичных разряда на **четные и нечетные**. Сеть со структурой **3-10-10-1**, все нейроны — сигмоидальные(гиперболический тангенс).

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-38).

Применение сетей прямого распространения (LXXIII)

Тестирование методов обучения – 6

Пример задачи классификации (2)

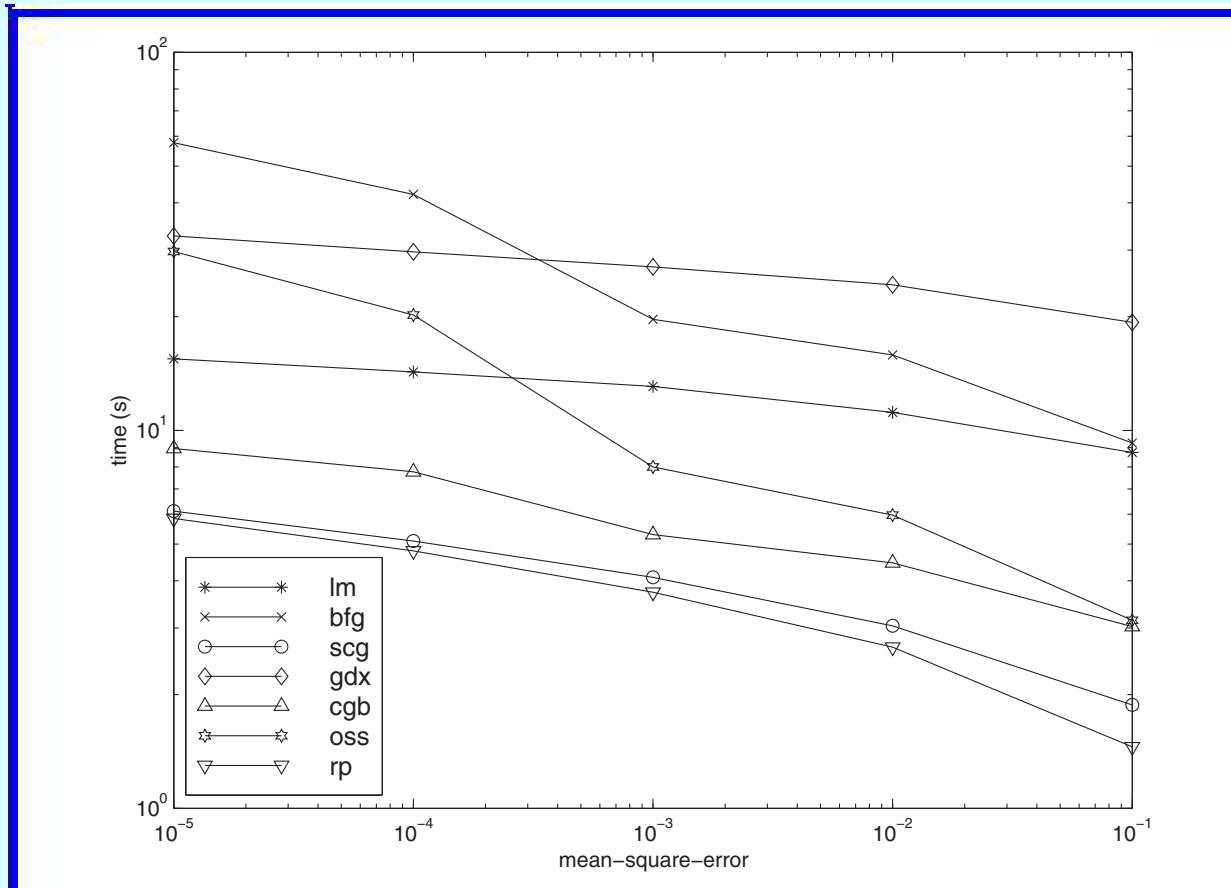


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-39).

Применение сетей прямого распространения (LXXIV)

Тестирование методов обучения – 7

Пример задачи классификации (3)



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 5-39).

Сети радиальных базисных функций (I)

Глобальная и локальная рецептивность сетей – 1

Многослойные сети на основе **сигмоидальных** элементов (нейронов), рассматривавшиеся до сих пор, обычно именуют как **многослойный персепtron** или **мультиперсепtron** (**MLP** – Multiple Layer Perceptron).

Характерная черта сигмоидальных MLP – **добавление/исключение** нового элемента или **корректировка** параметров сигмоиды в нем оказывает **глобальное воздействие** на свойства сети в целом.

Эта **глобальность** обусловлена тем, что в выход сети вносит вклад **каждый** нейрон сети за счет того, что **любой входной вектор** MLP вызывает «ненулевое» срабатывание **всех** нейронов сети.

Следствие «глобальности» MLP – **любое** изменение обучающего набора (добавление, изъятие или замена части обучающих примеров) требует **полного переобучения** сети.

Сети с архитектурой MLP – **только один из классов** многослойных сетей прямого распространения.

Сети радиальных базисных функций (II)

Глобальная и локальная рецептивность сетей – 2

Альтернативное направление — локально-рецептивные многослойные сети, в которых, в противоположность MLP, **добавление/исключение** нового элемента или **корректировка** его параметров оказывает **локальное действие** на свойства сети в целом.

Эта **локальность** обусловлена тем, что **для любого входного вектора** вносить вклад в выход локально-рецептивной сети будет лишь **ограниченное число** нейронов за счет того, что в таких сетях «ненулевое» срабатывание происходит **только для части** нейронов сети.

Следствие «локальности» сети — **любое** изменение обучающего набора (добавление, изъятие или замена части обучающих примеров) требует только **частичного переобучения** сети.

Сети радиальных базисных функций (III)

Предыстория – 1

Метод потенциальных функций

М. А. Айзerman, Э. М. Браверман, Л. И. Розоноэр (1968):

Айзerman М.А., Браверман Э.М., Розоноэр Л.И. Метод потенциальных функций в теории обучения машин. – М.: Наука, 1970. – 384 с.

Сети CMAC (Cerebellar Model Articulation Controller)

Джеймс Альбус (James Albus) (1975):

Albus J.S. A new aproach to manipulator control: The cerebellar model articulation controller (CMAC) // Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME, Series G. – 1975. – Vol. 97, No. 3. – p. 220–227.

Albus J.S. Data storage in the cerebellar model articulation controller (CMAC) // Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME, Series G. – 1975. – Vol. 97, No. 3. – p. 228–233.

Сети радиальных базисных функций (IV)

Предыстория – 2

Радиальные базисные функции (RBF)

Buhmann M.D. Radial basis functions: Theory and implementation. – Cambridge University Press, 2004. – 270 pp.

Сети радиальных базисных функций

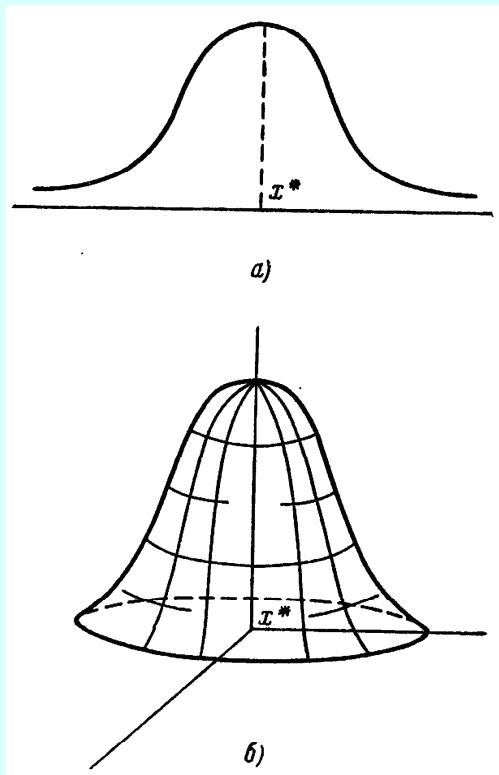
В современном варианте **RBF-сети** начинаются с работ:

Moody J., Darken C. Fast learning in networks of locally-tuned processing units // *Neural Computation*. – 1989. – No. 1. – pp. 281–294.

Poggio T., Girosi F. A theory of networks for approximation and learning. – Massachusetts Institute of Technology, Artificial Intelligence Laboratory, A.I. Memo No. 1140, July 1989. – 65 pp.

Сети радиальных базисных функций (V)

Метод потенциальных функций – 1



Потенциальная функция

$K(x, y)$ – функция двух переменных, x, y – точки из пространства X .

Пусть $y = x^*$ – некоторым образом **закрепленная точка** из X .

Тогда $K(x, x^*)$ – **функция точки** $x \in X$, зависящая от того, как выбрана точка $x^* \in X$.

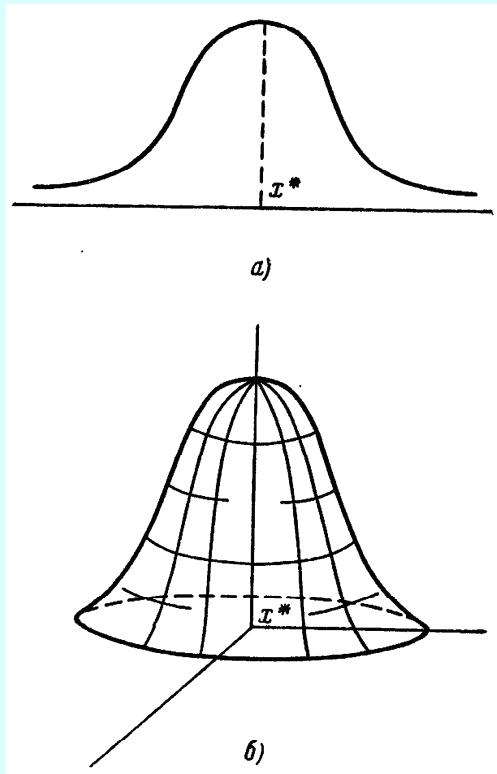
Пример подобной функции в физике – **потенциал**, определенный для любой точки пространства, но зависящий от того, где расположен *источник потенциала*.

По аналогии, функцию $K(x, y)$ называют **потенциальной функцией**.

Источник: Айзерман М.А., Браверман Э.М., Розонов Л.И. Метод потенциальных функций в теории обучения машин. – М.: Наука, 1970. – 384 с. (Рис. 5, с. 31).

Сети радиальных базисных функций (VI)

Метод потенциальных функций – 2



Свойства потенциальной функции

- функция $K(x, y)$ всюду положительна;
- она убывает при удалении точки x, y от точки $y = x^*$, при $x = x^*$ она достигает максимума.

Пусть K – функция расстояния $\rho(x, y)$ между точками x и y , т.е. $K = K[\rho(x, y)]$.

Функция $K(x, y)$ описывает поверхность – «холм» с вершиной над точкой $x = x^*$.

Примеры потенциальных функций:

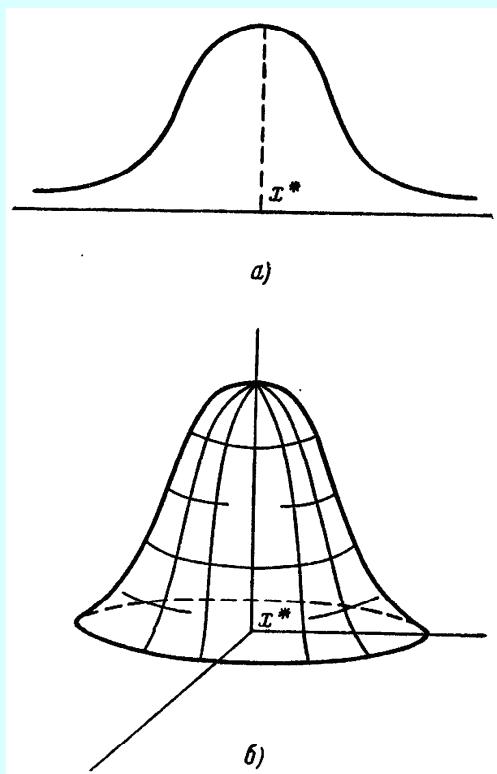
$$K(\rho) = e^{-\alpha \rho^2}$$

$$K(\rho) = \frac{1}{1 + \alpha \rho^2}$$

Здесь $\alpha = \text{const}$, $\alpha > 0$.

Сети радиальных базисных функций (VII)

Метод потенциальных функций – 3



Задача классификации (1)

Надо научиться относить точки к одному из двух классов (*A* и *B*).

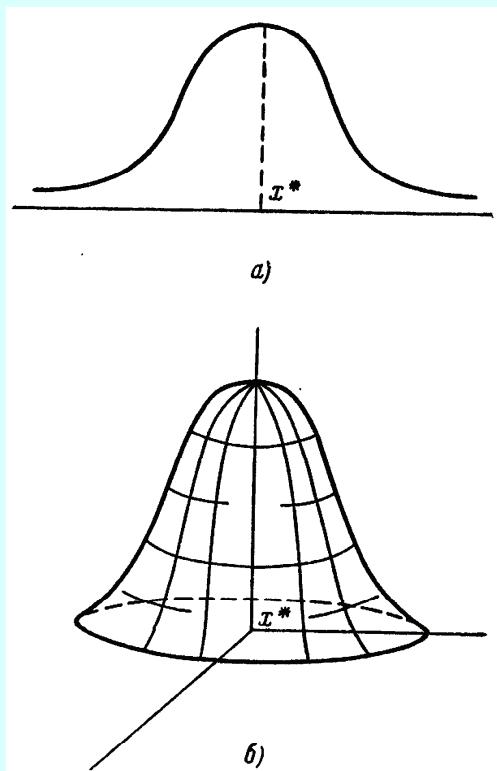
Пусть учителем показана точка $x = x^1$ и сообщено, что она принадлежит к классу *A*.

Примем точку $x = x^1$ за «источник потенциала», положив $x^* = x^1$, т.е. построим «холм» с вершиной в этой точке и запомним, что этот холм относится к точке из класса *A*.

Для следующих точек x^* из *A* или из *B* – аналогичные холмы с указанием на класс, к которому принадлежит каждый холм.

Сети радиальных базисных функций (VIII)

Метод потенциальных функций – 4



Задача классификации (2)

Суммарные потенциалы

Потенциалы, построенные над точками, принадлежащими классам **A** и **B**:

$$K_A(x) = \sum_{x^s \in A} K(x, x^s)$$

$$K_B(x) = \sum_{x^s \in B} K(x, x^s)$$

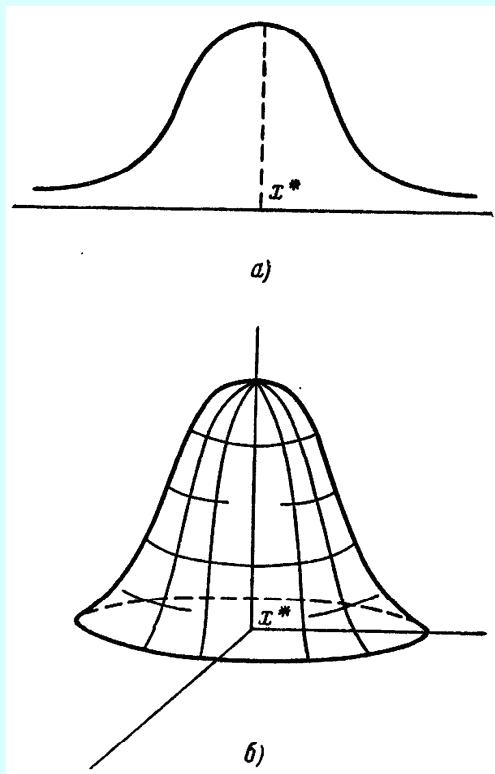
Холмы, построенные над точками $x \in A$, образуют в совокупности «гору» над областью, где расположены эти точки.

Аналогичная «гора» — для точек $x \in B$.

Результат: построены функции $K_A(x)$ и $K_B(x)$ — потенциалы образов **A** и **B**.

Сети радиальных базисных функций (IX)

Метод потенциальных функций – 5



Задача классификации (3)

Классификация новых паттернов

Потенциалы $K_A(x)$ и $K_B(x)$ как инструмент классификации.

Процедура классификации:

Точка $x = \tilde{x}$ относится к классу A , если

$$K_A(\tilde{x}) > K_B(\tilde{x})$$

и к классу B – в противном случае.

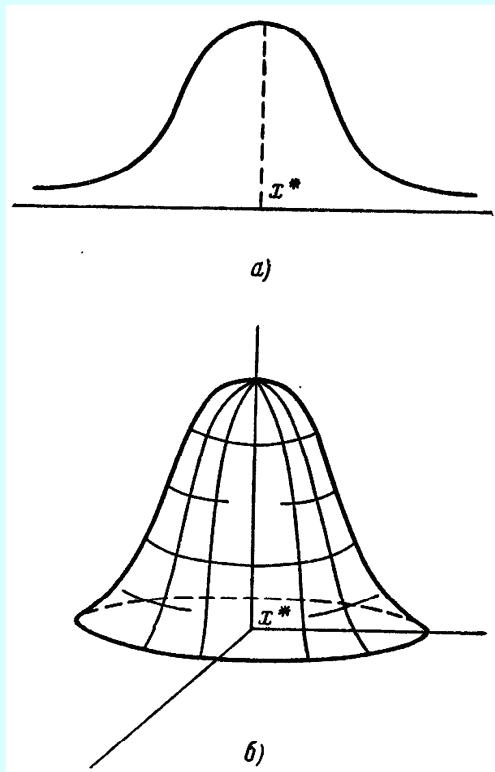
Разделяющая граница

$$\Phi(x) = K_A(x) - K_B(x)$$

Функция $\Phi(x)$ **положительна** над точками из A и **отрицательна** над точками из B , т.е. она **разделяет** знаком множества паттернов A и B .

Сети радиальных базисных функций (X)

Метод потенциальных функций – 6



Задача классификации (4)

Обучение без учителя

Случай, когда для точек $x \in X$ отсутствуют метки классов.

$$A \cup B = X, \quad A \cap B = \emptyset$$

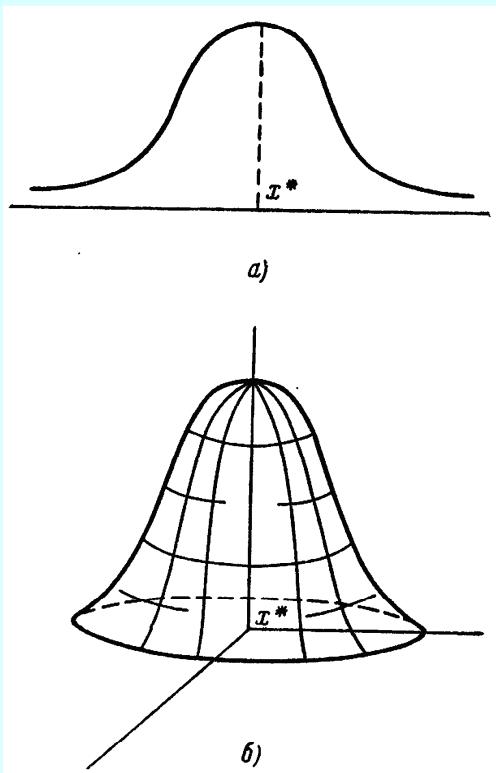
Области $D_A(x)$ и $D_B(x)$, которым принадлежат паттерны, относимые, соответственно, к классам A и B , **компактны**, т. е. по форме они «**достаточно просты**».

Отсутствие меток классов: невозможно отдельно выстраивать потенциалы $K_A(x)$ и $K_B(x)$.

Компактность $D_A(x)$ и $D_B(x)$: паттерны (точки $x \in X$) из классов A и B **группируются в кластеры**.

Сети радиальных базисных функций (XI)

Метод потенциальных функций – 7



Задача классификации (5)

Обучение без учителя

Общий потенциал для всех точек $x \in X$ без привязки их к классам:

$$\Phi(x) = \sum_{\forall x^s \in X} K(x, x^s)$$

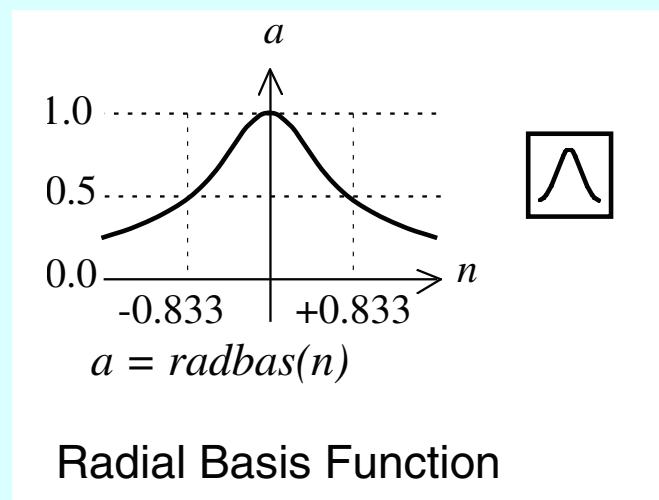
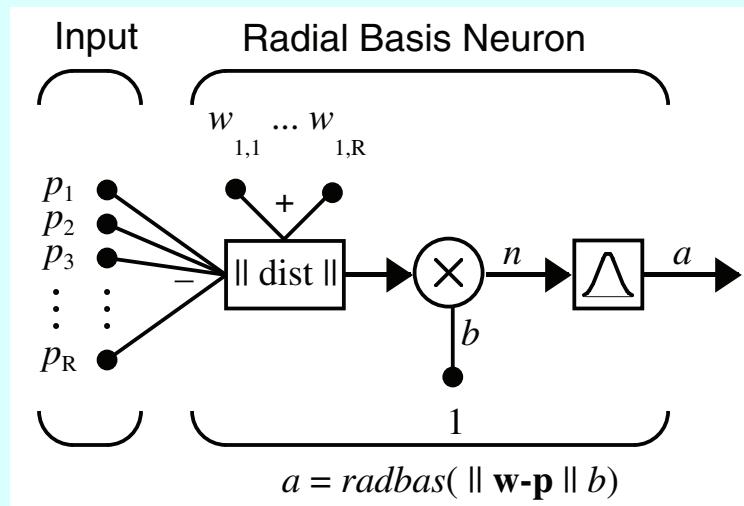
Функция $\Phi(x)$ представляет собой «горный ландшафт» с вершинами над областями A и B , с ущельями между ними.

Если найти «ущелья», т. е. поверхности минимума функции $\Phi(x)$, разделяющие вершины, то они будут **отделять области**, относящиеся к различным классам.

Процедура поиска **поверхностей минимума** функции $\Phi(x)$ – **обучение без учителя** методом потенциальных функций для решения задачи классификации.

Сети радиальных базисных функций (XI)

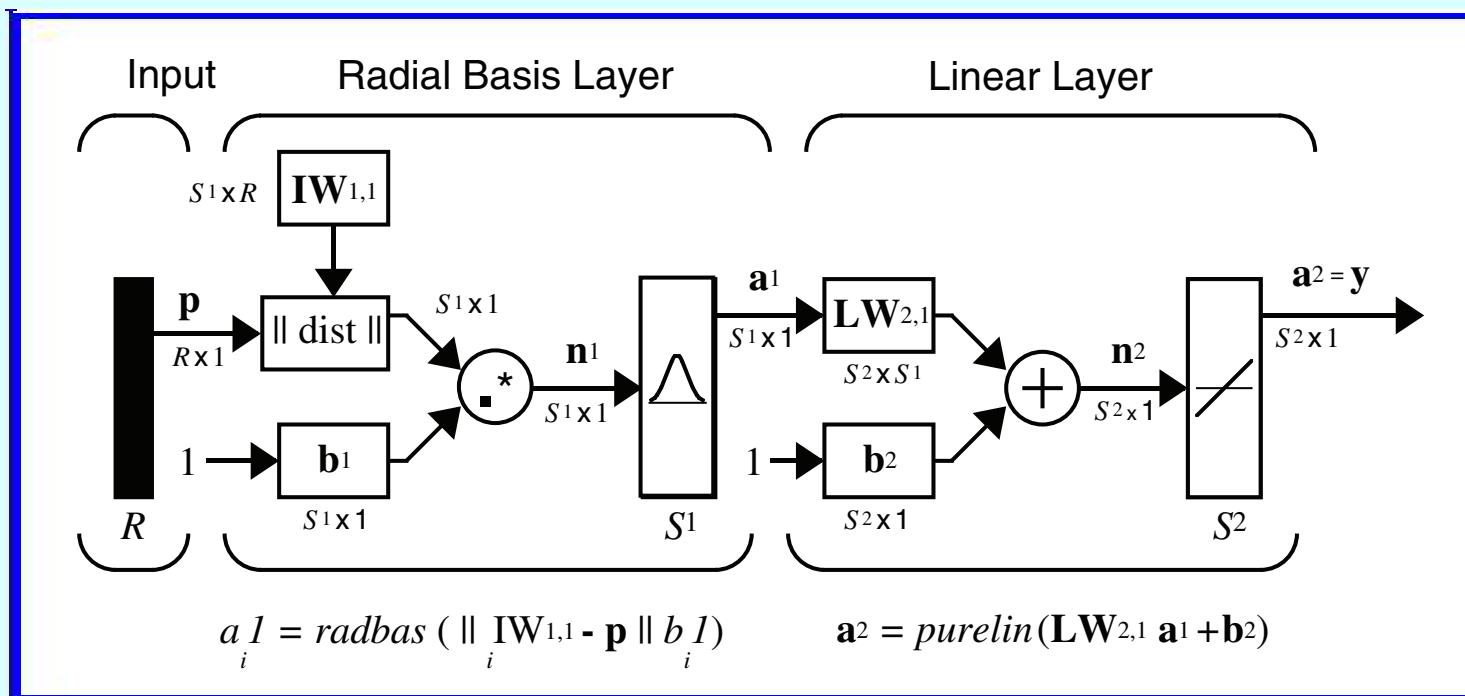
RBF-нейрон



Источник: *Demuth H., Beale M., Hagan M.*. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 8-3).

Сети радиальных базисных функций (XII)

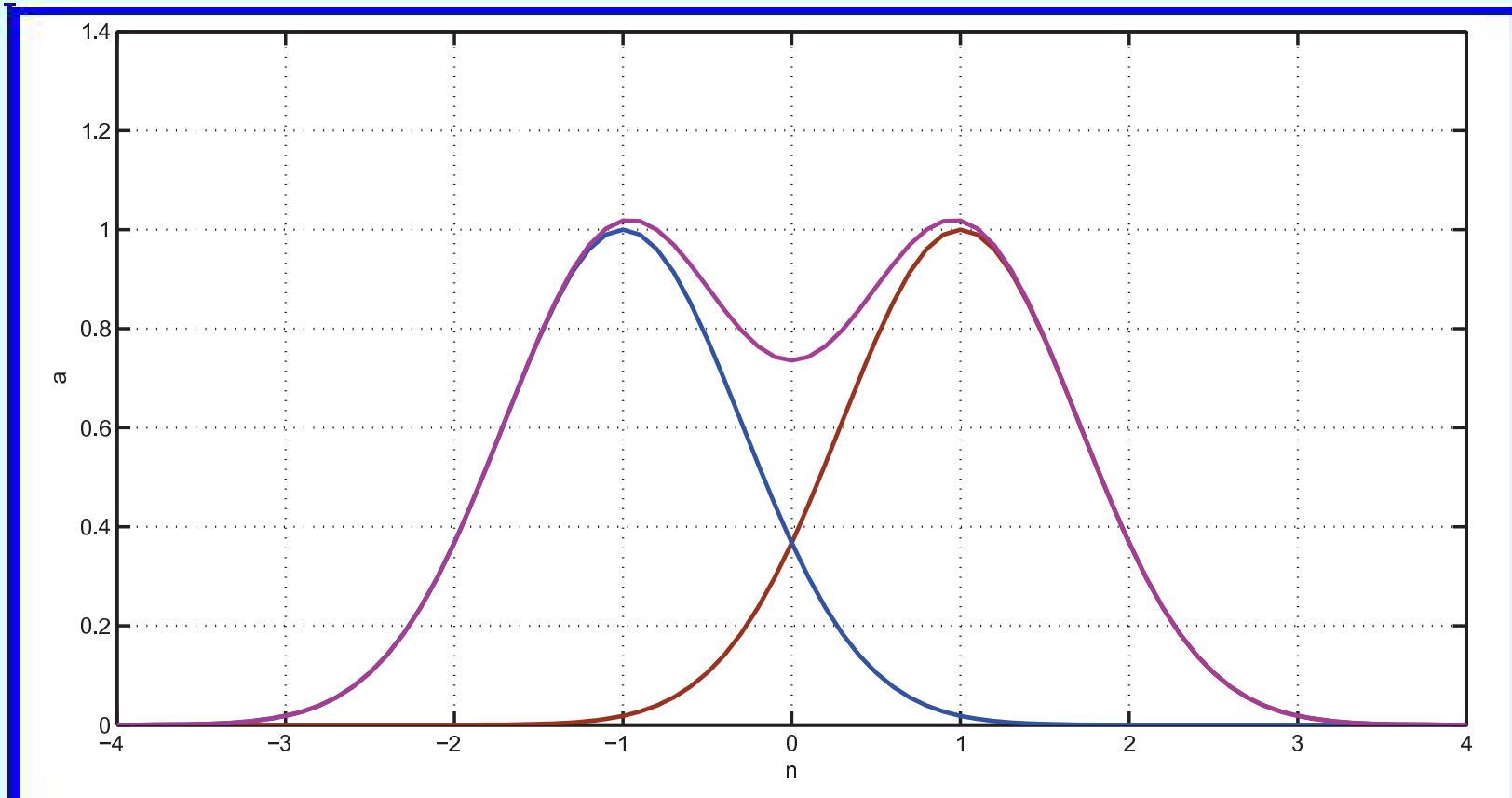
RBF-сеть



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 8-4).

Сети радиальных базисных функций (XIII)

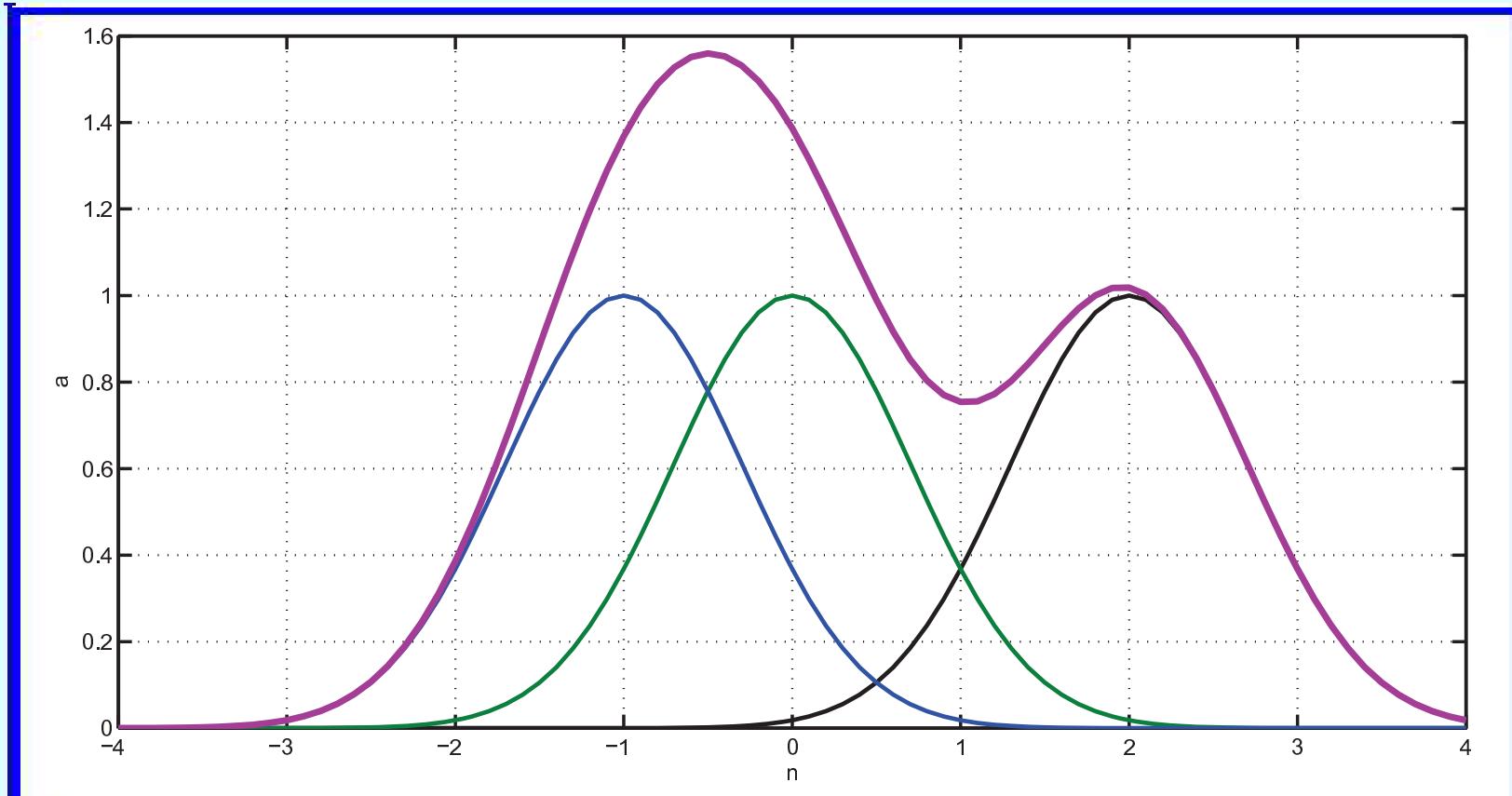
RBF и задача аппроксимации – 1



$$a = \text{radbas}(n + c) = e^{-(n+c)^2}$$

$c = 1$ — синяя линия; $c = -1$ — коричневая линия; **огибающая** — фиолетовая линия

Сети радиальных базисных функций (XIV) RBF и задача аппроксимации – 2

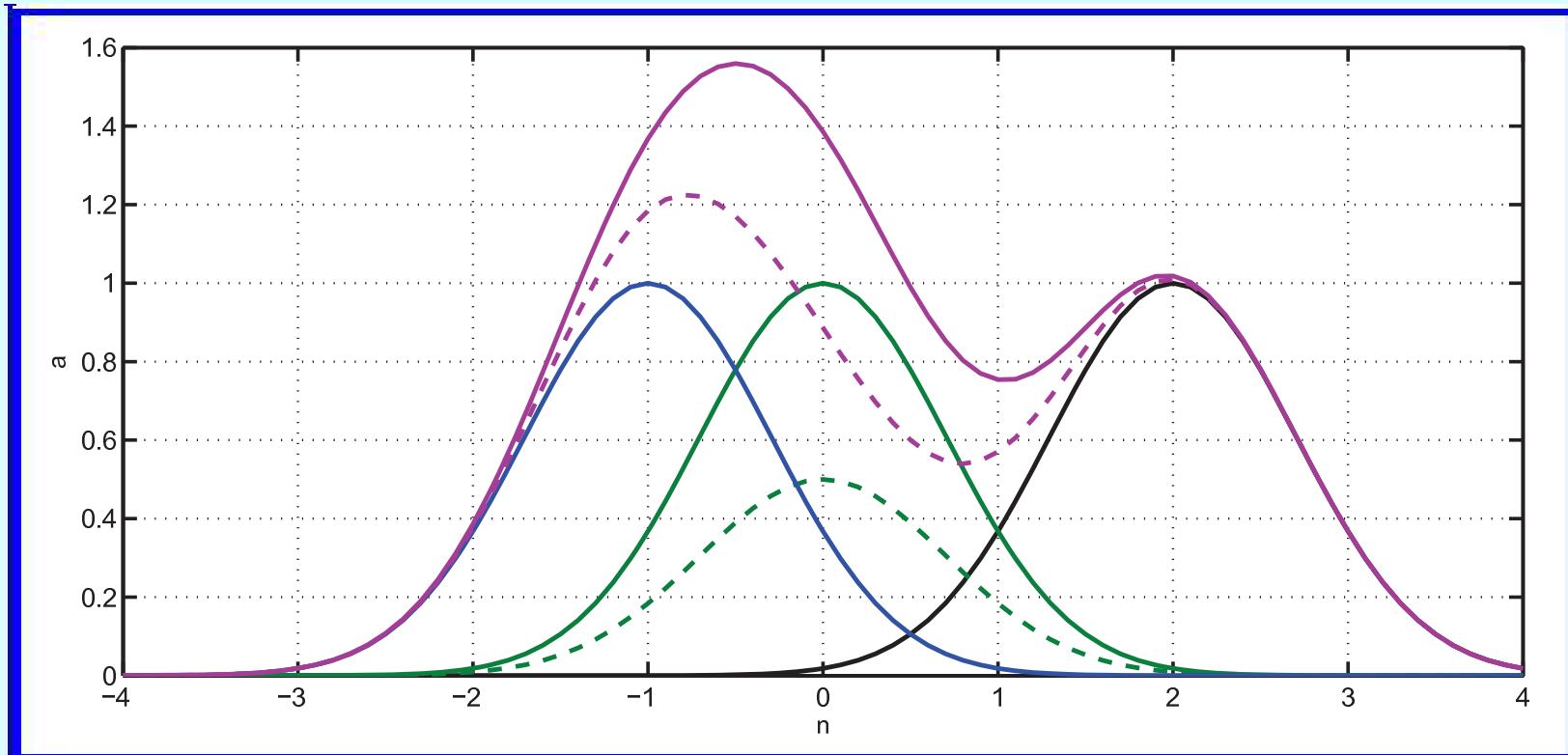


$$a = \text{radbas}(n + c) = e^{-(n+c)^2}$$

$c = 1$ — синяя линия; $c = 0$ — зеленая линия; $c = -2$ — черная линия;
огибающая — фиолетовая линия

Сети радиальных базисных функций (XV)

RBF и задача аппроксимации – 3

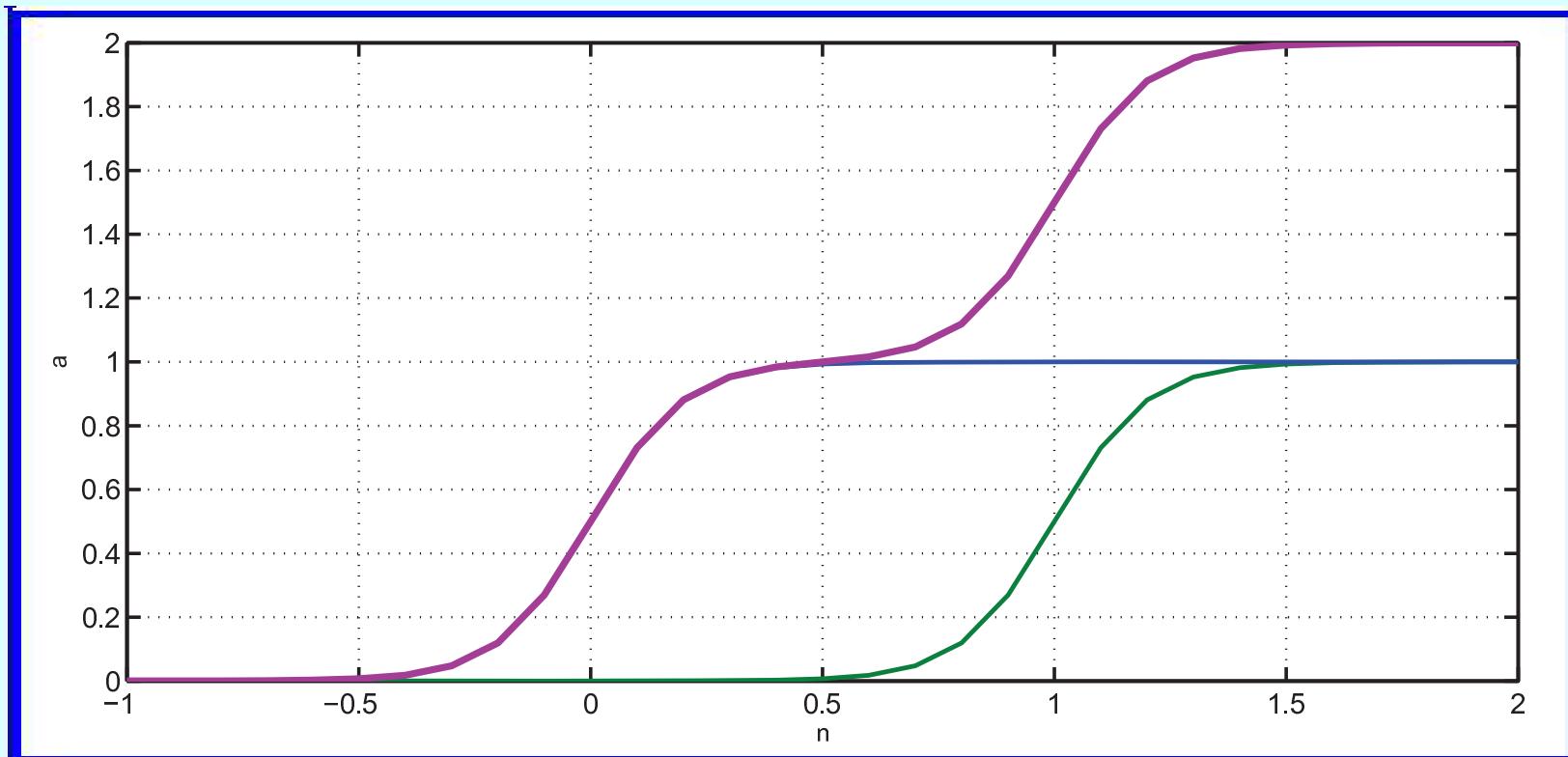


$$a = \text{radbas}(n + c) = e^{-(n+c)^2}$$

$c = 1$ — синяя линия; $c = 0$ — зеленая линия; $c = -2$ — черная линия;
огибающая — фиолетовая линия и *пунктирная* фиолетовая линия
(для $a = 0.5 \cdot \text{radbas}(n)$ при $c = 0$ — зеленый пунктир)

Сети радиальных базисных функций (XVI)

RBF-vs-Sigmoid – 1

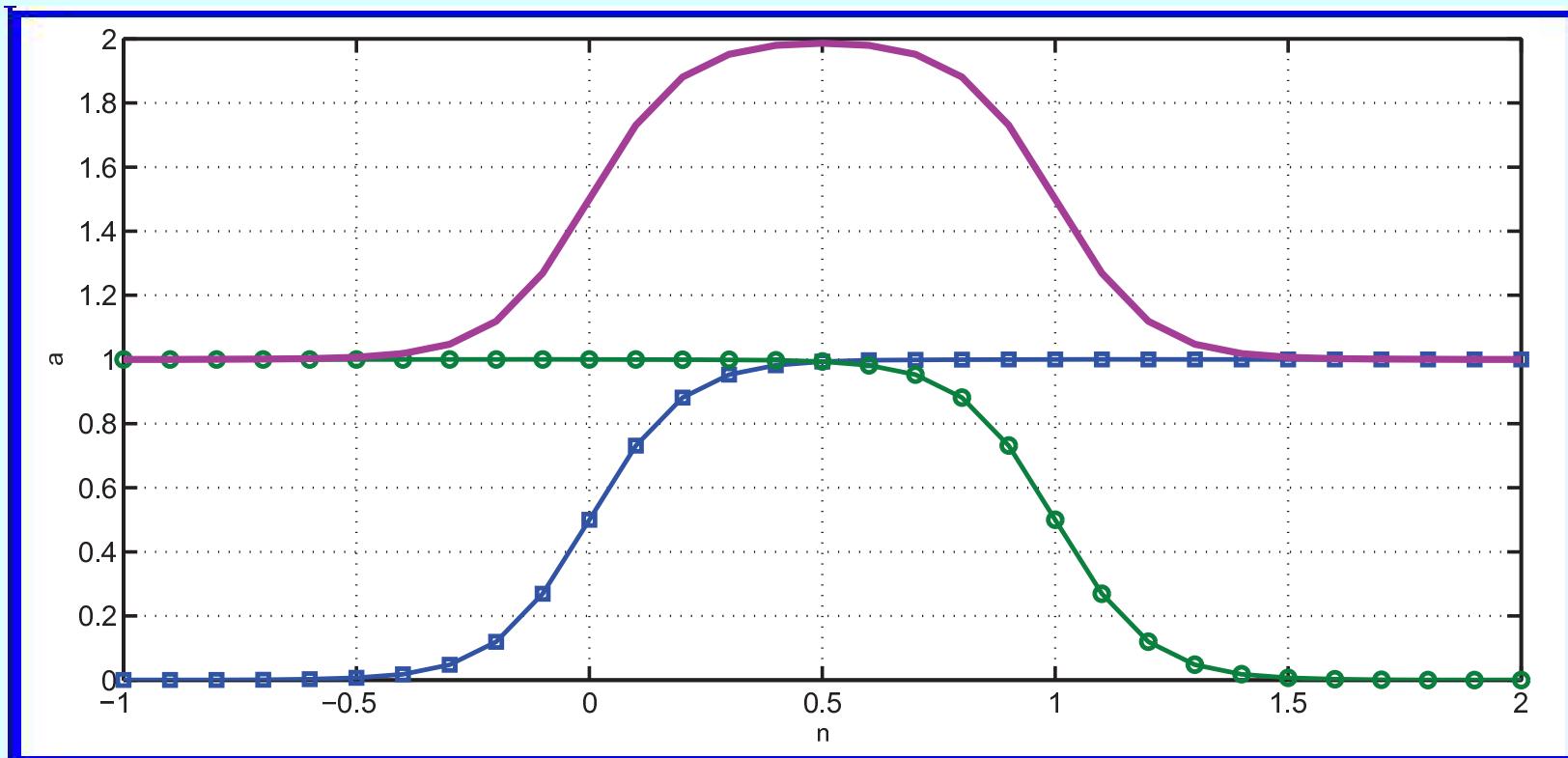


$$a = \text{logsig}(10 \cdot n + c) = 1 / (1 + e^{-(10 \cdot n + c)})$$

$c = 0$ — синяя линия; $c = -10$ — зеленая линия; **огибающая** — фиолетовая линия

Сети радиальных базисных функций (XVII)

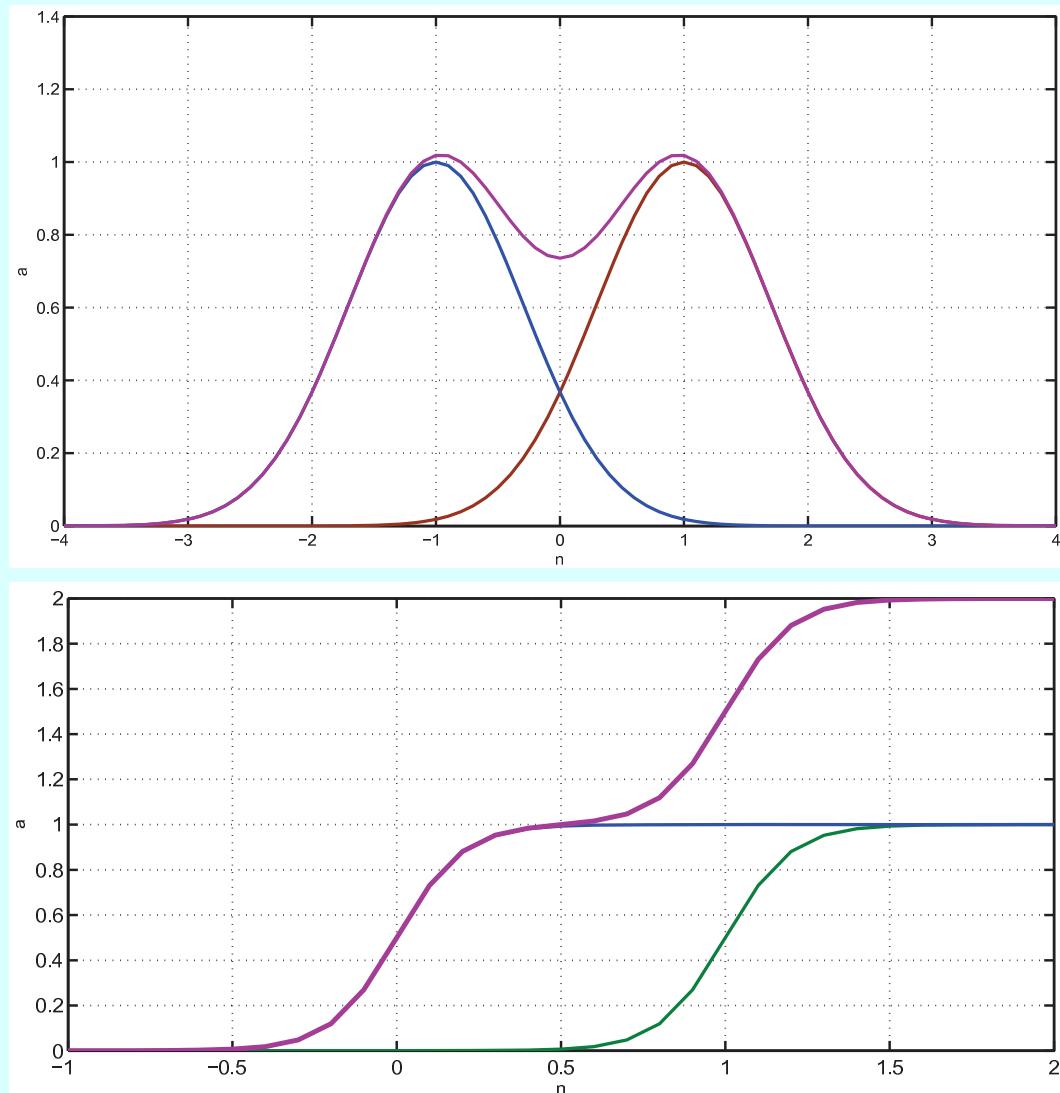
RBF-vs-Sigmoid – 2



$$a = \text{logsig}(10 \cdot n + c) = 1 / (1 + e^{(10 \cdot n + c)})$$

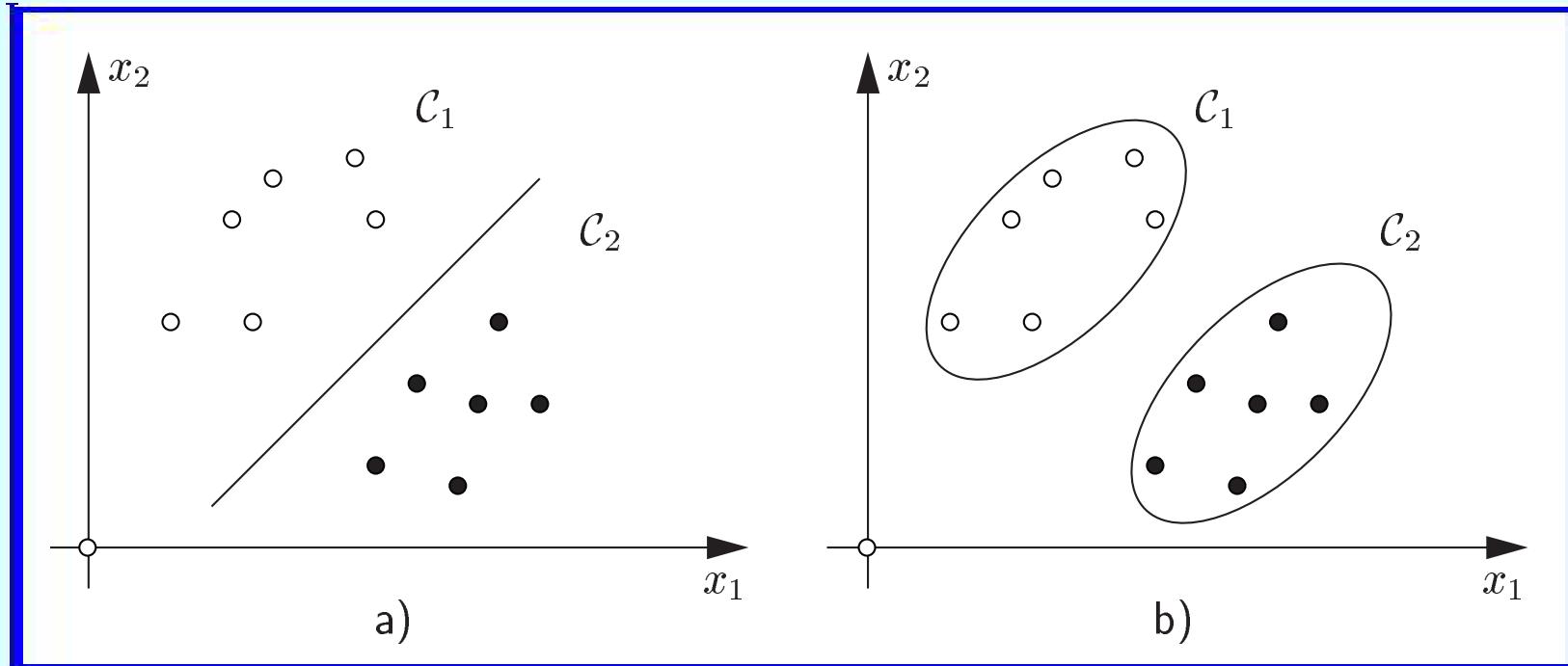
c = 0 — синяя линия; **c = -10** — зеленая линия; **огибающая** — фиолетовая линия

Сети радиальных базисных функций (XVIII) RBF-vs-Sigmoid – 3



Сети радиальных базисных функций (XIX)

RBF и задача классификации

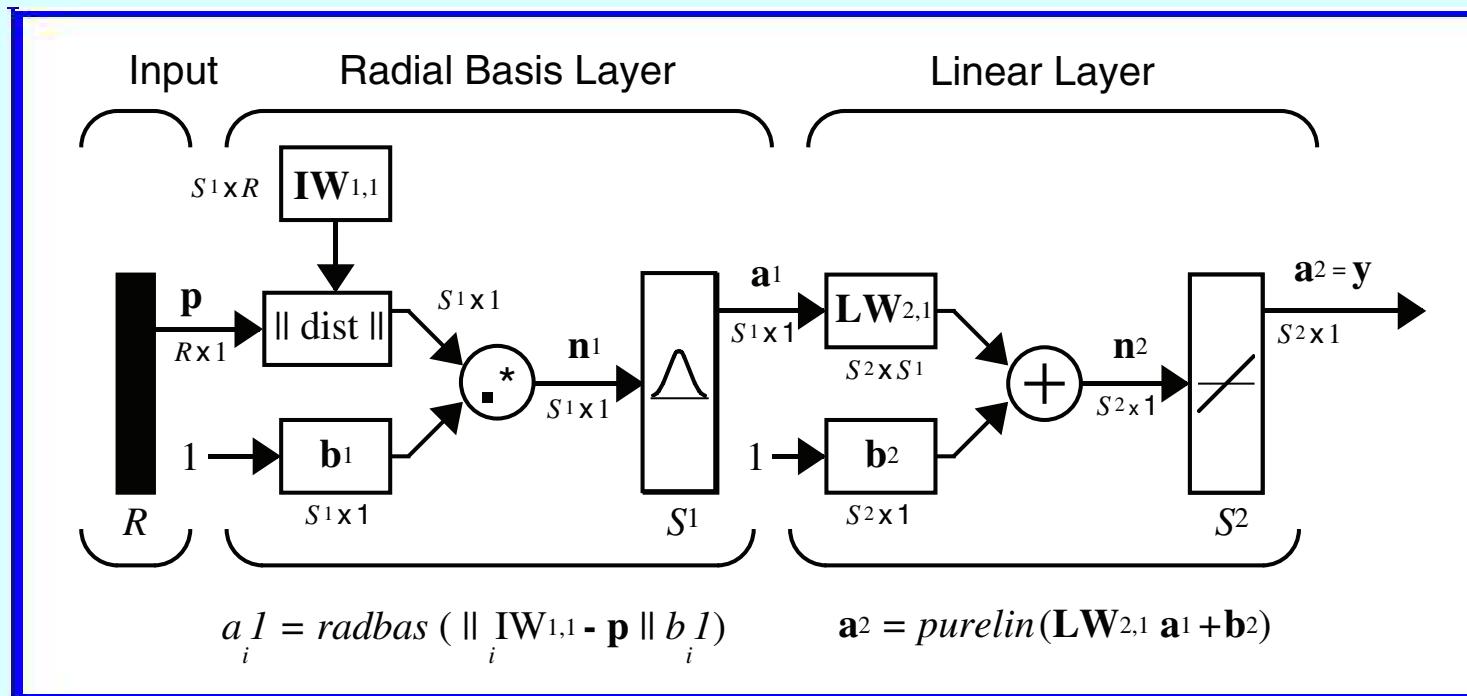


Классификация, выполненная персепtronом (a) и RBF-сетью (b)

Источник: *Hristev R. M.* The ANN book. – Electronic Edition. – 1998. – 388 pp.
(Figure 10.2, p. 179).

Сети радиальных базисных функций (ХХ)

RBF-сеть



$\| dist \|$ — вычисление расстояния между входным и весовым векторами

$(.*)$ — поэлементное умножение двух векторов

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 8-4).

Сети радиальных базисных функций (XXI)

Метрический классификатор – 1

Метрический классификатор (similarity-based classifier) — алгоритм классификации, основанный на вычислении **оценок сходства** между объектами.

Некоторые виды метрических классификаторов:

- метод ближайших соседей;
- метод потенциальных функций;
- метод радиальных базисных функций.

Понятие сходства формализуется через **функцию расстояния** $\rho(x, x')$ между объектами $x \in X$ и $x' \in X$.

Гипотеза компактности:

Схожие объекты, как правило, лежат в **одном классе**.

Это означает, что граница между классами имеет «достаточно простую форму», и **классы** образуют **компактно локализованные области** в пространстве объектов.

Сети радиальных базисных функций (ХХII)

Метрический классификатор – 2

Связь с рассуждениями по прецедентам

(CBR — Case-Based Reasoning)

Действия алгоритма метрической классификации

Вопрос:

Почему объект $u \in X$ отнесен к классу $y \in Y$?

Ответ:

Потому, что имеются схожие с ним прецеденты (объекты) класса y .

В ответе предъявляется **список** схожих объектов.

Сети радиальных базисных функций (XXIII)

Метрический классификатор – 3

$A : X \rightarrow Y$ — алгоритм классификации
 X — объекты; Y — ответы (имена классов)

Обучающая выборка как набор пар «объект-ответ»:

$$X^m = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$$

$\rho(x, x') : X \times X \rightarrow [0, \infty)$ — функция расстояния
как мера сходства двух объектов

Чем больше значение функции $\rho(x, x')$, тем менее схожими являются два объекта $x \in X$ и $x' \in X$.

Объекты обучающей выборки $x_i \in X^m$, упорядоченные в порядке возрастания расстояния до некоторого произвольного объекта u :

$$\rho(u, x_{1,u}) \leq \rho(u, x_{2,u}) \leq \dots \leq \rho(u, x_{k,u}) \leq \dots \leq \rho(u, x_{m,u})$$

$x_{i,u}$ — объект обучающей выборки, являющийся i -м соседом объекта u .

$y_{i,u}$ — ответ (имя класса) для i -го соседа объекта u .

Произвольный объект u порождает свою перенумерацию выборки.

Сети радиальных базисных функций (XXIV)

Метрический классификатор – 4

Метрический алгоритм классификации с обучающей выборкой X^m относит объект $u \in X$ к тому классу $y \in Y$, для которого суммарный вес ближайших обучающих объектов $\Gamma_y(u, X^m)$ максимальен:

$$a(u, X^m) = \operatorname{argmax}_{y \in Y} \underbrace{\sum_{i=1}^m [y_{i,u} = y] w(i, u)}_{\Gamma_y(u, X^m)}$$

$w(i, u)$ — степень важности i -го соседа для классификации объекта u

$\Gamma_y(u, X^m)$ — оценка близости объекта u к классу y

Сети радиальных базисных функций (XXV)

Метрический классификатор – 5

Методы ближайших соседей:

- метод ближайшего соседа;
- метод k ближайших соседей;
- метод взвешенных ближайших соседей.

Метод ближайшего соседа: Классифицируемый объект x относят к тому классу y_i , которому принадлежит **ближайший сосед** x_i из обучающей выборки X .

Метод k ближайших соседей: Классифицируемый объект x относят к тому классу y_i , которому принадлежит **большинство из k** его ближайших соседей.

Метод взвешенных ближайших соседей: Каждому i -му соседу приписывается вес w_i . Объект относят к тому классу, который набирает **наибольший суммарный вес** среди k ближайших соседей.

Сети радиальных базисных функций (XXVI)

Метрический классификатор – 6

Метод ближайшего соседа

(nearest neighbor)

$$\rho(u, x_{1,u}) \leq \rho(u, x_{2,u}) \leq \dots \leq \rho(u, x_{k,u}) \leq \dots \leq \rho(u, x_{m,u})$$

Классифицируемый объект x относят к тому классу y_i , которому принадлежит ближайший сосед x_i из обучающей выборки X .

$$w(i, u) = [i = 1]; \quad a(u, X^m) = y_{1,u}$$

Сети радиальных базисных функций (XXVII)

Метрический классификатор – 7

Метод k ближайших соседей

(k nearest neighbors, k-NN)

$$\rho(u, x_{1,u}) \leq \rho(u, x_{2,u}) \leq \dots \leq \rho(u, x_{k,u}) \leq \dots \leq \rho(u, x_{m,u})$$

Классифицируемый объект x относят к тому классу y_i , которому принадлежит
большинство из k его ближайших соседей.

$$w(i, u) = [i \leq k]; \quad a(u, X^m) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^m [y_{i,u} = y]$$

Сети радиальных базисных функций (XXVIII)

Метрический классификатор – 8

Метод взвешенных ближайших соседей

(weighted k nearest neighbors)

$$\rho(u, x_{1,u}) \leq \rho(u, x_{2,u}) \leq \dots \leq \rho(u, x_{k,u}) \leq \dots \leq \rho(u, x_{m,u})$$

Каждому i -му соседу приписывается вес w_i . Объект относят к тому классу, который набирает **наибольший суммарный вес** среди k ближайших соседей.

$$w(i, u) = [i \leq k] w_i; \quad a(u, X^m) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^m [y_{i,u} = y] w_i$$

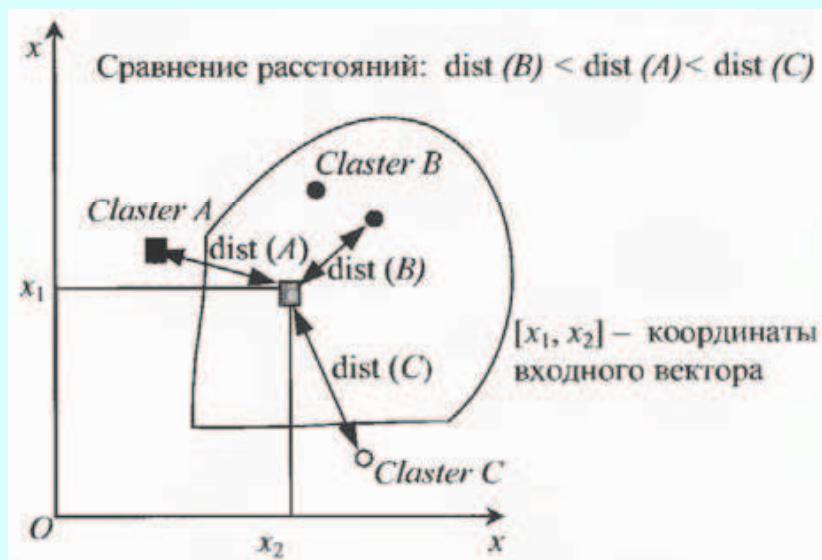
w_i — вес, зависящий только от номера соседа

$w_i = \frac{k+1-i}{k}$ — **линейно убывающие веса**

$w_i = q^i, 0 < q < 1$ — **экспоненциально убывающие веса**

Сети радиальных базисных функций (XXIX)

Метрический классификатор – 9



**Классификация по алгоритму
 k ближайших соседей**

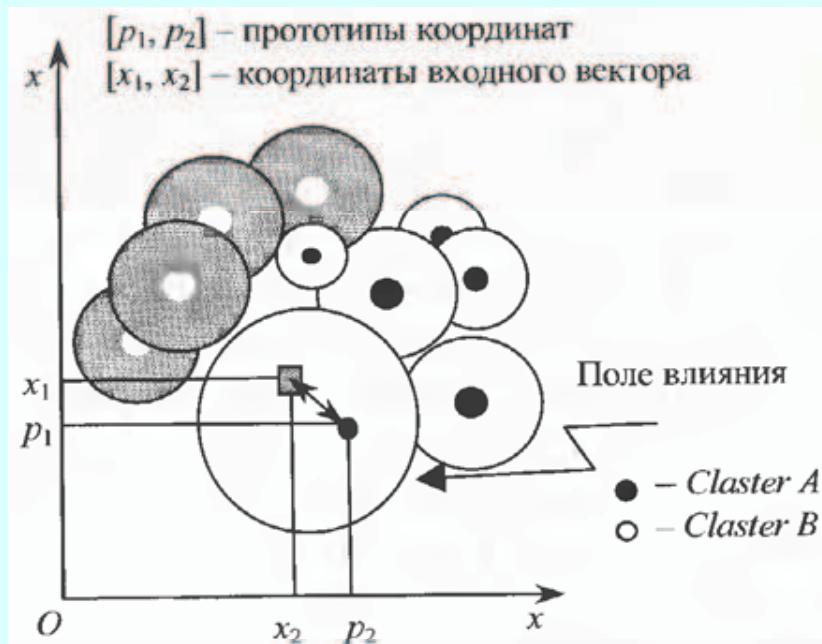
$$w(i, u) = [i \leq k];$$

$$a(u, X^m) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^m [y_{i,u} = y]$$

Источник: Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 9.2, с. 230).

Сети радиальных базисных функций (XXX)

Метрический классификатор – 10



Представление прототипов и их полей
влияния в двумерном пространстве

Поле влияния — часть N -мерного пространства вокруг прототипа, где возможно обобщение.

Классифицирующее решение:

- если вводимый вектор не лежит ни в каком поле влияния, он **не распознается**;
- если вводимый вектор лежит в поле влияния одного или более прототипов, относящихся к одному классу, он объявляется **принадлежащим этому классу**;
- если вводимый вектор лежит в поле влияния двух или более прототипов, относящихся к различным классам, он объявляется **узнанным, но формально не определенным**.

Источник: Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 9.3, с. 231).

Сети радиальных базисных функций (XXXI)

Метрический классификатор – 11

Классификация

Пример k -NN классификации

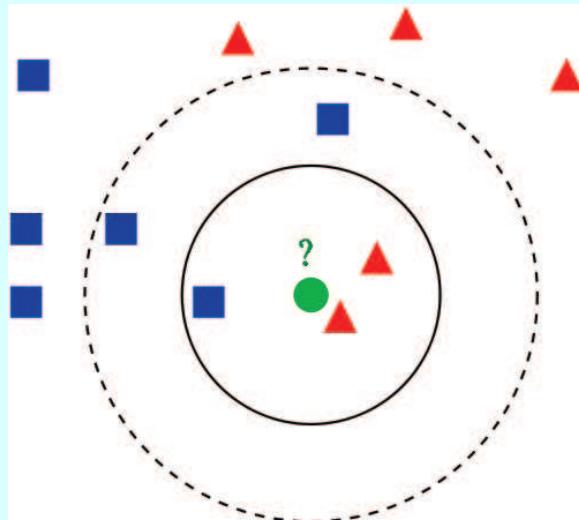
Зеленый кружок — тестовый пример

Синий квадрат — объект из класса 1

Красный треугольник — объект из класса 2

Если $k = 3$ — тестовый пример будет отнесен к **классу 2** (ближайшие **соседи** — 2 треугольника и 1 квадрат).

Если $k = 5$ — тестовый пример будет отнесен к **классу 1** (ближайшие **соседи** — 2 треугольника и 3 квадрата).



Источник: k -nearest neighbor algorithm. – From Wikipedia.

Сети радиальных базисных функций (XXXII)

Метрический классификатор – 12

Метод потенциальных функций

$$\rho(u, x_{1,u}) \leq \rho(u, x_{2,u}) \leq \dots \leq \rho(u, x_{k,u}) \leq \dots \leq \rho(u, x_{m,u})$$

$$w(i, u) = \gamma_{i,u} K\left(\frac{\rho(u, x_{i,u})}{h_{i,u}}\right)$$

$$a(u, X^m) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^m [y_i = y] \gamma_{i,u} K\left(\frac{\rho(u, x_i)}{h_i}\right)$$

γ_i — веса объектов, $\gamma_i > 0$, $h_i > 0$

Физическая аналогия:

γ_i — величина заряда в точке x_i

h_i — радиус действия потенциала с центром в точке x_i

y_i — знак заряда (предполагается, что $Y = \{-1, +1\}$)

$K(r) = \frac{1}{r}$ — в электростатике

Сети радиальных базисных функций (XXXIII)

Стратегии обучения – 1

Варианты стратегий обучения:

- случайный выбор** фиксированных центров;
- выбор центров **на основе самоорганизации**;
- выбор центров **с учителем**.

Сети радиальных базисных функций (XXXIV)

Стратегии обучения – 2

Случайный выбор фиксированных центров (1)

Простейший подход — использование **фиксированных RBF** в качестве активационных функций скрытого слоя сети.

Размещение центров — случайным образом (**случайная выборка** из множества имеющихся примеров).

Гауссовская функция в качестве активационной:

$$G(||\mathbf{x} - \mathbf{t}_i||^2) = \exp\left(\frac{m}{d_{max}^2} ||\mathbf{x} - \mathbf{t}_i||^2\right), \quad i = 1, 2, \dots, m$$

Здесь **m** — количество центров; **d_{max}** — максимальное расстояние между выбранными центрами.

Все RBF имеют **фиксированную ширину**:

$$\sigma = \frac{d_{max}}{\sqrt{2m}}$$

Это гарантирует, что RBF не будут слишком «остроконечными» или слишком «расплющенными».

Сети радиальных базисных функций (XXXV)

Стратегии обучения – 3

Случайный выбор фиксированных центров (2)

Настраиваемые параметры — синаптические веса w выходного слоя.

Настройка — метод псевдообращения:

$$w = G^+ d$$

d — вектор **желаемого отклика** для множества примеров

Матрица G^+ является **псевдообратной** матрице G :

$$G = \{g_{ji}\}$$

$$g_{ji} = \exp\left(-\frac{m}{d_{max}^2} \|x_j - t_i\|^2\right), \quad i = 1, 2, \dots, m; \quad j = 1, 2, \dots, N$$

x_j — это **j-й входной вектор** из множества обучающих примеров

Основной недостаток данной стратегии обучения — требуется большое число обучающих примеров.

Сети радиальных базисных функций (XXXVI)

Стратегии обучения – 4

Выбор центров на основе самоорганизации

Для снижения уровня требований к объему задачника — **гибридный процесс обучения**, состоящий из двух этапов:

- этап обучения на основе самоорганизации** для нахождения подходящих положений центров RBF;
- этап обучения с учителем**, на котором находятся значения весов выходного слоя.

Этап обучения на основе самоорганизации — алгоритм кластеризации.

Этап обучения с учителем — алгоритм LMS.

Сети радиальных базисных функций (XXXVII)

Стратегии обучения – 5

Выбор центров с учителем

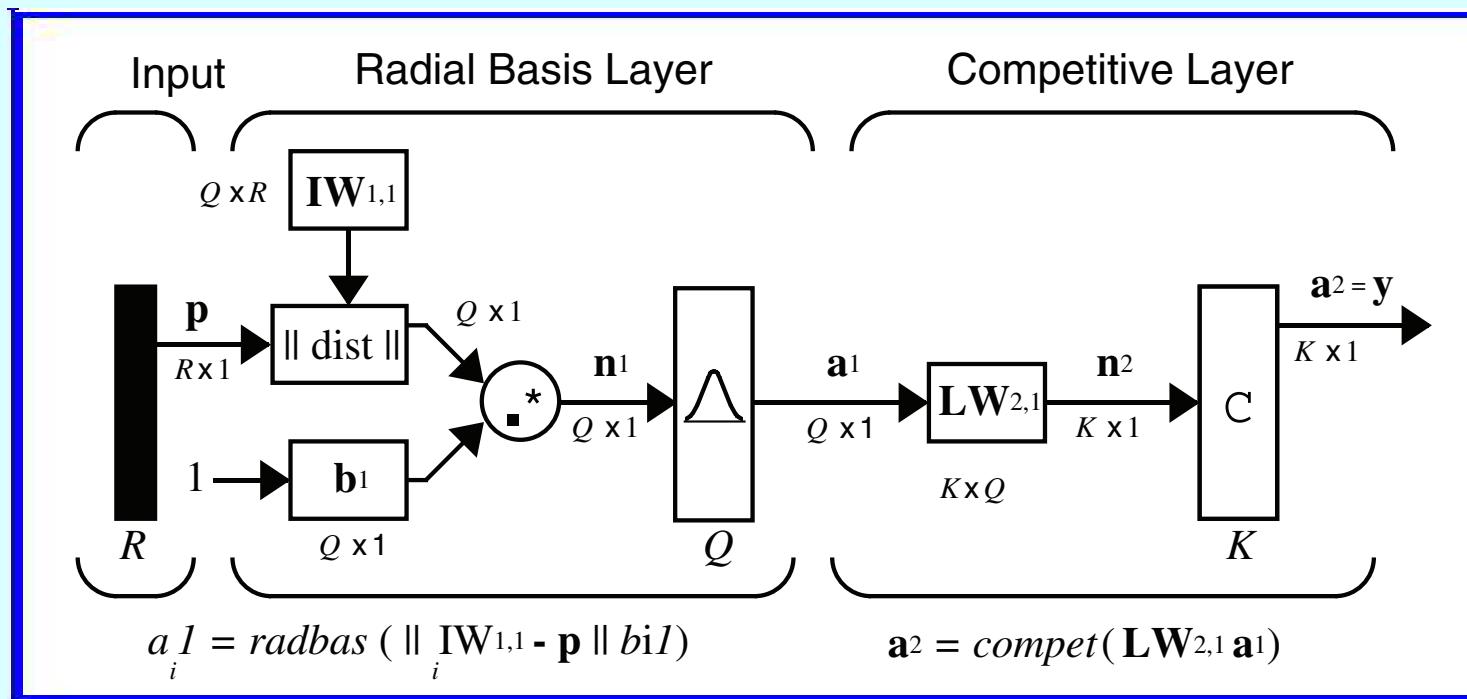
Настраиваются:

- веса** выходного слоя;
- положение центров RBF** для скрытого слоя;
- ширина RBF** в скрытом слое.

Минимизация функции ошибки – **обобщенный алгоритм LMS**.

Сети радиальных базисных функций (XXXVIII)

PNN-сеть

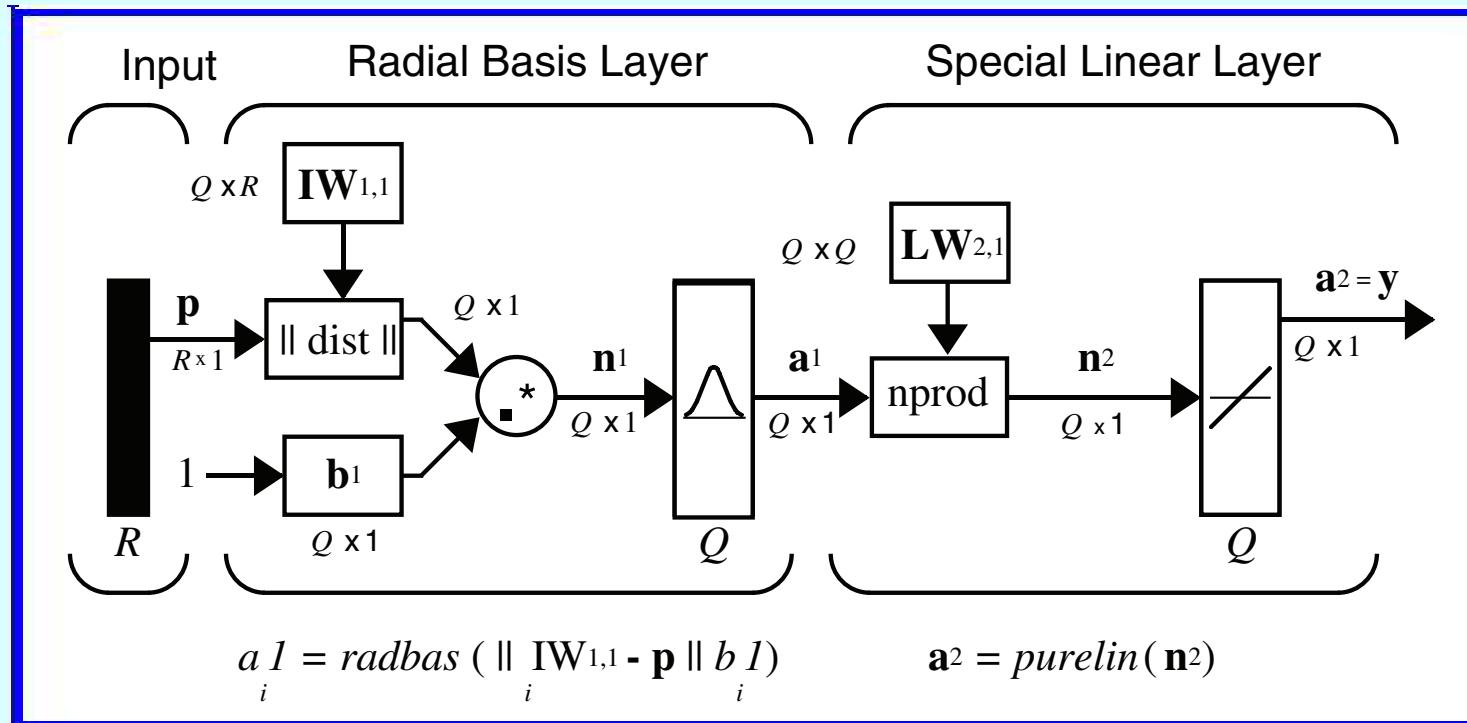


PNN — Probabilistic Neural Network

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 8-9).

Сети радиальных базисных функций (XXXIX)

GRNN-сеть



GRNN — Generalized Regression Neural Network

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 8-12).

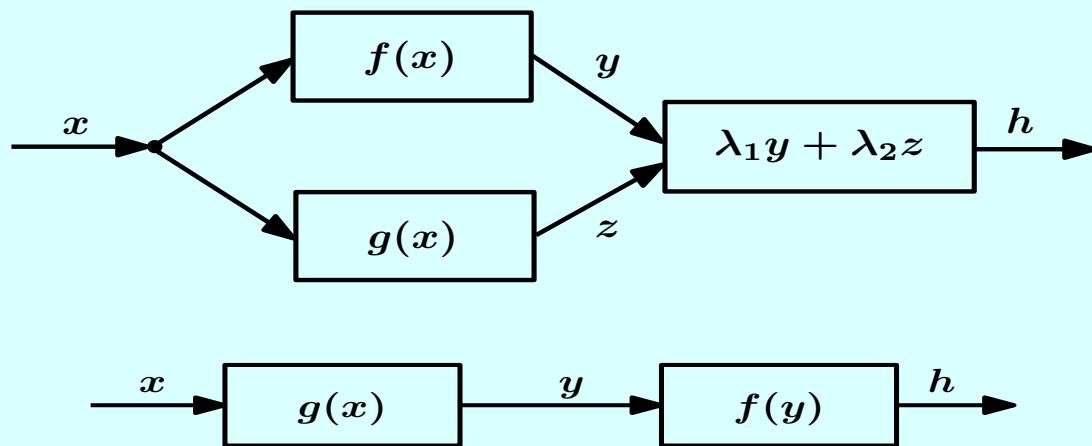
Нейросети и составное представление функций (I)

О составном представлении функций – 1

Нейросети как **составные отображения** строятся на базе «строительных блоков» двух видов.

В первом из вариантов отдельные функции можно суммировать, перемножать и т. п., то есть выполнять над ними **арифметические операции**.

Например, для функций $y = f(x)$ и $z = g(x)$ можно определить **сумму** вида $s(x) = \lambda_1 f(x) + \lambda_2 g(x)$, как это показано на верхнем рисунке.



Во втором из вариантов выход одной функции подается на вход другой — таким образом реализуется **композиция** функций (отображений).

Например, для функций $y = g(x)$ и $z = f(y)$ можно определить **композицию** вида $s(x) = (g \circ f)(x) = f(g(x))$, как это показано на нижнем рисунке.

Нейросети и составное представление функций (II)

О составном представлении функций – 2

Если трактовать нейросети как различные комбинации «строительных блоков», показанных на предыдущем слайде, то возникает **ряд вполне резонных вопросов**:

- 1) Неужели ничего **похожего** на НС-представления не было?
- 2) В чем же состоят **отличия** НС-варианта от «классики»?
- 3) В чем **сила** НС-подхода и каковы пределы его возможностей?

Эти три вопроса **принципиально важны** не только для НС-моделирования, но и для вычислительного моделирования в целом.

Попытаемся найти **ответы** на эти вопросы.

Нейросети и составное представление функций (III)

Разложения по системам функций – 1

Рассмотрим **первый из вопросов**, сформулированных только что:

- 1) Неужели ничего **похожего** на НС-представления не было?
- 2) В чем же состоят **отличия** НС-варианта от «классики»?
- 3) В чем **сила** НС-подхода и каковы пределы его возможностей?

Ответ на этот вопрос — было. Такие подходы **существуют**.

Они основаны на различных **разложениях** по некоторой системе функций.

Нейросети и составное представление функций (IV)

Разложения по системам функций – 2

Итак, ответ на вопрос о том, **есть ли** составные представления функций, похожие на НС-модели — да, такие представления **существуют**.

Они основаны на различных **разложениях** по некоторой системе функций.

В достаточно общем виде такое разложение можно представить в виде:

$$f(x) = \sum_{i=1}^n \lambda_i \varphi_i(x),$$

где λ_i — весовые коэффициенты.

Если вспомнить введенные выше «строительные блоки» для формирования составных представлений функций, то, очевидно, мы имеем здесь дело с **первым** из этих блоков, который осуществляет **арифметические операции с функциями-примитивами** $\varphi_i(x)$ (в данном случае — умножение каждой из функций на скалярный вес λ_i и их суммирование).

Нейросети и составное представление функций (V)

Разложения по системам функций – 3

Вспомним теперь **адаптивный сумматор** — составную часть искусственного нейрона:

$$V = b + w_1 x_1 + \dots + w_N x_N = b + \sum_{i=1}^N w_i x_i .$$

Мы видим, что адаптивный сумматор («сжимающее отображение») представляет собой **специальный вид разложения по системе функций**, в данном случае — по системе простейших линейных функций x_i .

Таким образом, **традиционное разложение** по системе функций в определенном смысле «**проще**» искусственного нейрона (нет второго «строительного блока» — композиции функций), но, в то же время, «**сложнее**» (разложение осуществляется по системе нелинейных функций $\varphi_i(x)$, а не простейших линейных x_i).

Нейросети и составное представление функций (VI)

Разложения по системам функций – 4

Разложение по системе функций — подход, часто используемый в традиционной вычислительной математике.

В частности, разложение по системе функций лежит в основе решения задач интерполяции с помощью **полиномов Лагранжа**.

Если брать не готовый результат в виде интерполяционного полинома Лагранжа, приводимый обычно в учебниках по численному анализу, а посмотреть, **каким образом он получен**, обнаруживается следующее.

Нейросети и составное представление функций (VII)

Разложения по системам функций – 5

Среди способов интерполяции наиболее распространен случай **линейной интерполяции**, когда приближение отыскивается в виде:

$$g(x; a_1, \dots, a_n) = \sum_{i=1}^n a_i \varphi_i(x),$$

где $\varphi_i(x)$ – фиксированные функции, а значения коэффициентов a_i определяются из условия совпадения с приближаемой функцией в узлах интерполяции x_j

$$f(x_j) = \sum_{i=1}^n a_i \varphi_i(x_j), \quad j = 1, \dots, n. \quad (1)$$

Нейросети и составное представление функций (VIII)

Разложения по системам функций – 6

Наиболее изучен **случай интерполяции многочленами** вида

$$\sum_{i=1}^n a_i x^{i-1}, \quad (2)$$

Тогда

$$\varphi_i(x) = x^{i-1},$$

и система уравнений (1) имеет вид

$$\sum_{i=1}^n a_i (x^{i-1}) = f(x_j), \quad j = 1, \dots, n. \quad (3)$$

Здесь предполагается, что все x_j различны.

Нейросети и составное представление функций (IX)

Разложения по системам функций – 7

Мы получили здесь **систему линейных алгебраических уравнений**, неизвестными величинами в которой являются коэффициенты a_i . Можно пытаться решать систему (3) непосредственно, но это **не слишком хорошая идея**. Уже при сравнительно небольших n , например, при $n = 20$ происходит **катастрофическое искажение** коэффициентов a_i вычислительной погрешностью.

По этой причине стараются **найти более приемлемую** в вычислительном отношении **форму решения** задачи интерполяции с помощью многочленов.

Одной из таких форм и является **многочлен Лагранжа**.

Нейросети и составное представление функций (X)

Разложения по системам функций – 8

Итак, то, что было сказано об интерполяционных полиномах, в частности, о полиномах Лагранжа — **правда, но не вся правда**.

На самом деле, подходы к приближенному представлению зависимостей на основе интерполяционных полиномов и на основе разложений по системе функций **нельзя противопоставлять** друг другу.

Идейные корни у этих двух подходов **одни и те же**.

Нейросети и составное представление функций (XI)

Разложения по системам функций – 9

Многочленными представлениями далеко **не исчерпываются** существующие варианты разложений по системе функций, а задачами аппроксимации функций не исчерпывается перечень решаемых проблем.

Рассмотрим **более сложный пример**, иллюстрирующий потенциал данного подхода.

Нейросети и составное представление функций (XII)

Разложения по системам функций – 10

Пусть дано дифференциальное уравнение второго порядка

$$F(x, y, y', y'') = 0. \quad (4)$$

Двухточечная краевая задача для уравнения (4) ставится следующим образом: найти функцию $y = y(x)$, которая внутри отрезка $[a, b]$ удовлетворяет уравнению (4), а на концах отрезка — **краевым условиям**

$$\begin{aligned} \varphi_1[y(a), y'(a),] &= 0, \\ \varphi_2[y(b), y'(b),] &= 0 \end{aligned} \quad (5)$$

Пусть уравнение (4) и граничные условия (5) линейны. Тогда получаем **линейную краевую задачу**. Для нее дифференциальное уравнение и краевые условия записываются в виде:

$$y'' + p(x)y' + q(x)y = f(x), \quad (6)$$

$$\begin{aligned} \alpha_0y(a) + \alpha_1y'(a) &= A, \\ \beta_0y(b) + \beta_1y'(b) &= B, \end{aligned} \quad (7)$$

где $p(x)$, $q(x)$, $f(x)$ — известные непрерывные на отрезке $[a, b]$ функции, α_0 , α_1 , β_0 , β_1 , A , B — заданные постоянные, причем $\alpha_0 + \alpha_1 \neq 0$ и $\beta_0 + \beta_1 \neq 0$.

Нейросети и составное представление функций (XIII)

Разложения по системам функций – 11

Пусть на отрезке $[a, b]$ задана **система базисных функций**:

$$u_0(x), u_1(x), \dots, u_n(x), \dots, \quad (8)$$

удовлетворяющая следующим **условиям**.

1. Система (8) является **ортогональной**, т. е.

$$\begin{aligned} \int_a^b u_i(x)u_j(x)dx &= 0 && \text{при } i \neq j, \\ \int_a^b u_i^2(x)dx &\neq 0. \end{aligned} \quad (9)$$

2. Система (8) является **полной**, т. е. не существует никакой другой отличной от нуля функции, ортогональной ко всем функциям $u_i(x)$ ($i = 0, 1, 2, \dots$).

3. Конечная система базисных функций $\{u_i(x)\}$ ($i = 0, 1, \dots, n$) выбирается так, чтобы входящие в нее функции удовлетворяли **краевым условиям** (7).

Нейросети и составное представление функций (XIV)

Разложения по системам функций – 12

Решение краевой задачи (6), (7) будем искать в виде:

$$y(x) = u_0(x) + \sum_{i=1}^n c_i u_i(x)$$

Такое представление искомого решения (записываемого хотя и приближенным, но **аналитическим** выражением, а не таблицей чисел, как в разностных методах) используется, например, в **методе Галеркина**.

Нейросети и составное представление функций (XV)

Разложения по системам функций – 13

Пример. Для того, чтобы найти методом Галеркина приближенное аналитическое решение уравнения

$$y'' - y' \cos x + y \sin x = \sin x$$

удовлетворяющее **краевым условиям**

$$y(-\pi) = y(\pi) = 2$$

можно выбрать в качестве системы **базисных функций** $u_i(x)$ ($i = 1, 2, 3, 4$) следующие **тригонометрические функции**:

$$u_0 = 2, u_1 = \sin x, u_2 = \cos x + 1, u_3 = \sin 2x, u_4 = \cos 2x - 1.$$

Эти функции **линейно независимы** на отрезке $[-\pi, \pi]$.

Нейросети и составное представление функций (XVI)

Сопоставление нейросетевого подхода с «классикой» – 1

Рассмотрим теперь **второй из вопросов**, сформулированных ранее:

- 1) Неужели ничего **похожего** на НС-представления не было?
- 2) В чем же состоят **отличия** НС-варианта от «классики»?
- 3) В чем **сила** НС-подхода и каковы пределы его возможностей?

Нейросети и составное представление функций (XVII)

Сопоставление нейросетевого подхода с «классикой» – 2

Чтобы обоснованно ответить на вопрос **о сходстве и различиях** между НС-моделями и «классикой», необходимо построить достаточно **общую схему**, которая давала бы **базу** для такого сопоставления.

Такого рода схемой является понятие **композиционной модели**.

Основываясь на том, что уже было сказано о **составном представлении** зависимостей между величинами, введем это понятие таким образом, чтобы оно, по-возможности, **охватывало бы как традиционные, так и нейросетевые модели** приближенного описания зависимостей (как **«сверхзадача»** — и другие модели из области новых информационных технологий, о которых речь впереди).

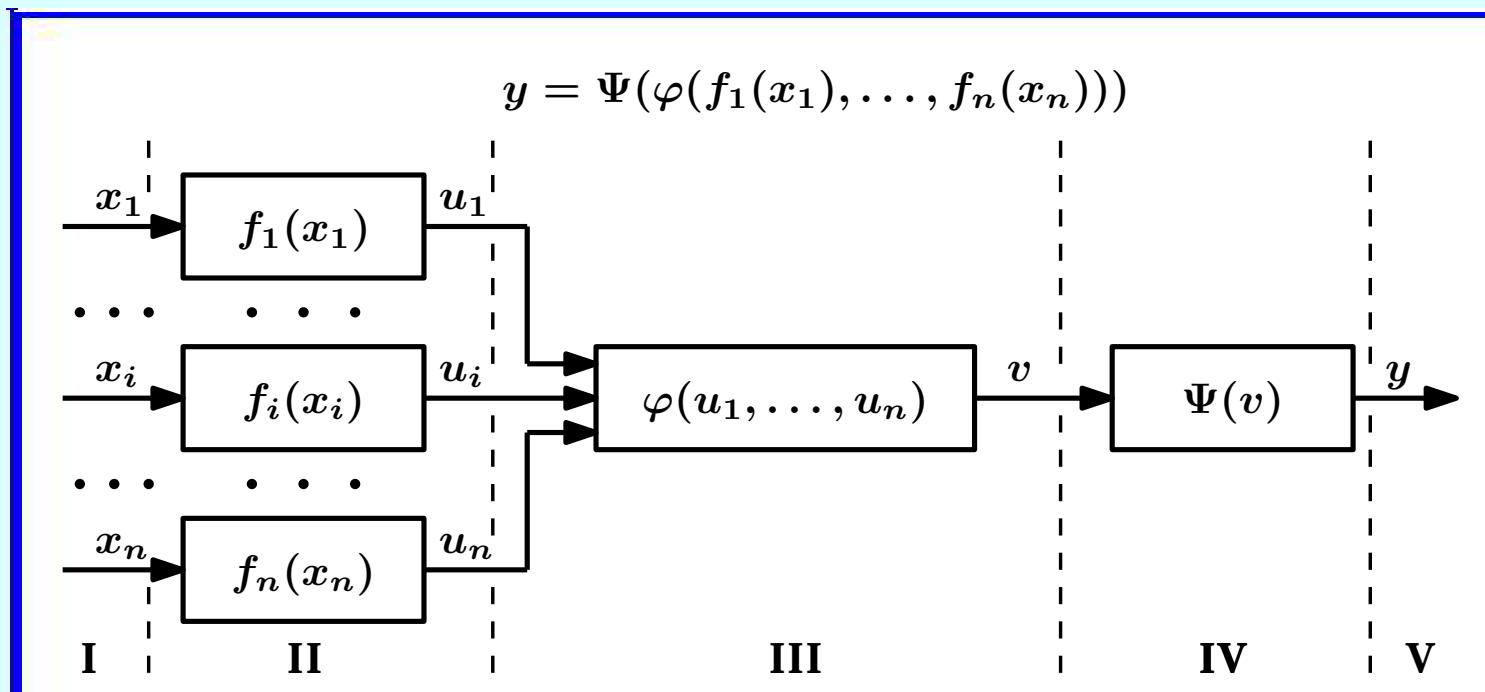
Как будет видно далее, композиционная модель представляет собой **сетевое объединение** различного рода **базисных модулей**.

Варьируя вид базисных модулей и характер связей между ними, можно получить **все многообразие** композиционных моделей.

Нейросети и составное представление функций (XVIII)

Сопоставление нейросетевого подхода с «классикой» – 3

Структура базисного модуля композиционной модели
в наиболее общем виде



I — входные объекты (исходные данные); **II** — входные отображения;
III — сжимающее отображение; **IV** — выходное отображение;
V — выходной объект (результат).

Нейросети и составное представление функций (XIX)

Сопоставление нейросетевого подхода с «классикой» – 4

Базисный модуль КМ общего вида реализует следующее **отображение**:

$$y = \Psi(\varphi(f_1(x_1), \dots, f_n(x_n))) \quad (10)$$

или, в обобщенной форме,

$$\Psi : X_1 \times X_2 \times \dots \times X_i \times \dots \times X_n \rightarrow Y \quad (11)$$

Его можно представить как **цепочку преобразований**:

$$\begin{aligned} & \langle x_1, \dots, x_n \rangle \Rightarrow \\ & \Rightarrow \langle f_1(x_1), \dots, f_n(x_n) \rangle \Rightarrow \\ & \Rightarrow \varphi(f_1(x_1), \dots, f_n(x_n)) \Rightarrow \\ & \Rightarrow \Psi(\varphi(f_1(x_1), \dots, f_n(x_n))) \Rightarrow y \end{aligned} \quad (12)$$

Здесь $x_1 \in X_1, \dots, x_n \in X_n, y \in Y$.

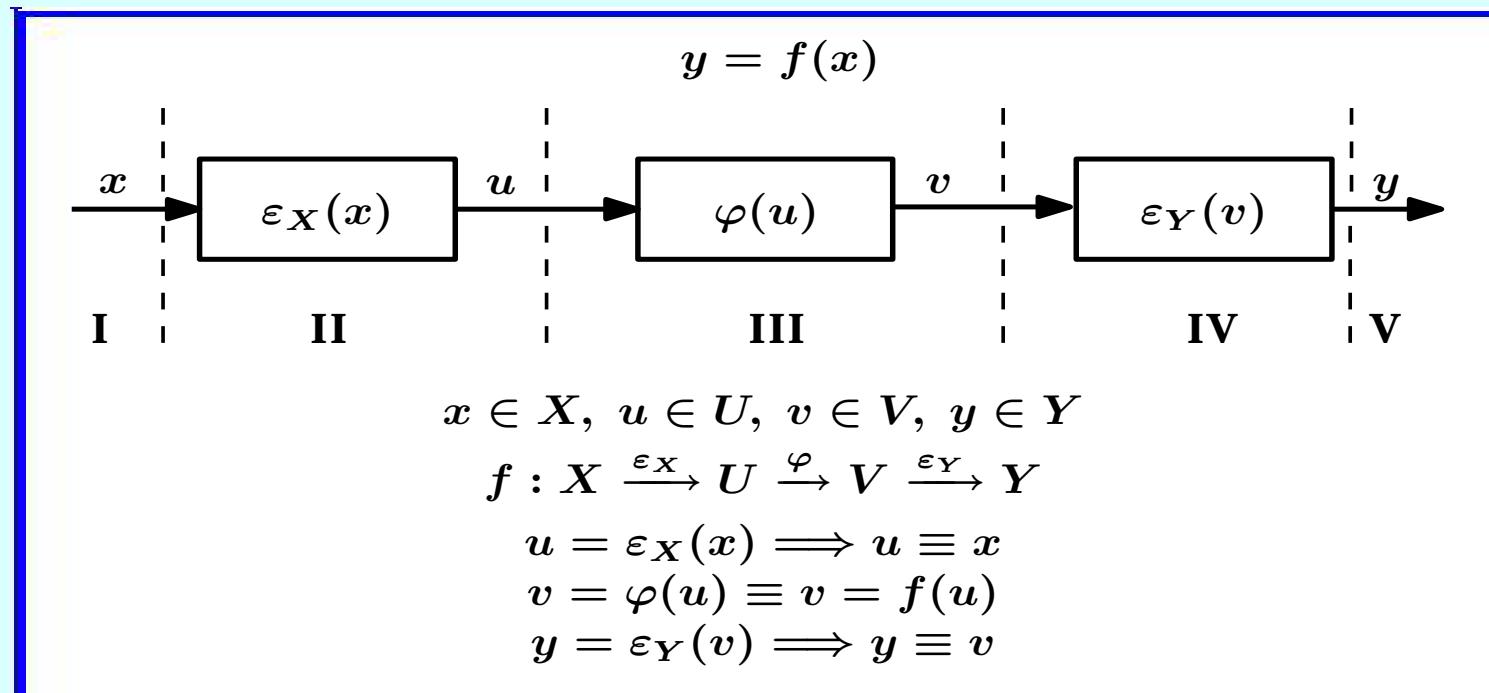
Варьируя вид отображений f_i, φ, Ψ , входящих в цепочку преобразований (12), получим базисные модули КМ с различными характеристиками.

Продемонстрируем этот факт на нескольких **характерных примерах**.

Нейросети и составное представление функций (XX)

Сопоставление нейросетевого подхода с «классикой» – 5

Конкретизация базисного модуля композиционной модели –
функция одной переменной

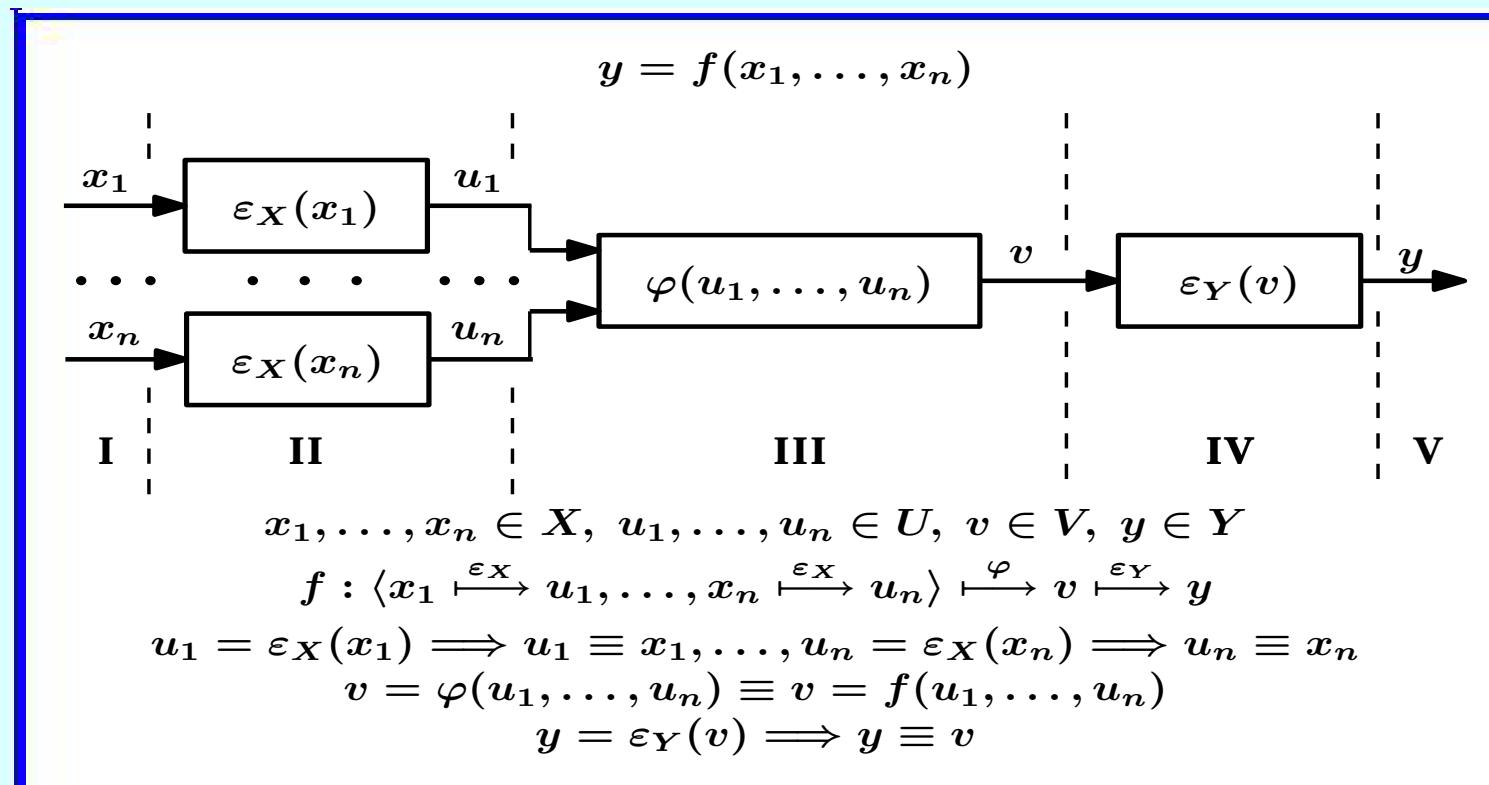


- I – входные объекты (исходные данные); II – входные отображения;
- III – сжимающее отображение; IV – выходное отображение;
- V – выходной объект (результат)

Нейросети и составное представление функций (XXI)

Сопоставление нейросетевого подхода с «классикой» – 6

Конкретизация базисного модуля композиционной модели –
функция нескольких переменных



I – входные объекты (исходные данные); **II** – входные отображения;
III – сжимающее отображение; **IV** – выходное отображение;
V – выходной объект (результат).

Нейросети и составное представление функций (XXII)

Сопоставление нейросетевого подхода с «классикой» – 7

Элемент нейронной сети прямого распространения

Исходные данные:

$$\mathbf{x} = (x_1, \dots, x_n)$$

Входные отображения:

$$f_i(x_i) \Rightarrow \mathbf{x}_i = u_i$$

Сжимающее отображение:

$$\varphi(u_1, \dots, u_n) \Rightarrow \sum_{i=0}^n w_i u_i = v$$

Выходное отображение:

$$\Psi(v) \Rightarrow \frac{1}{1 + e^{-Tv}} = y$$

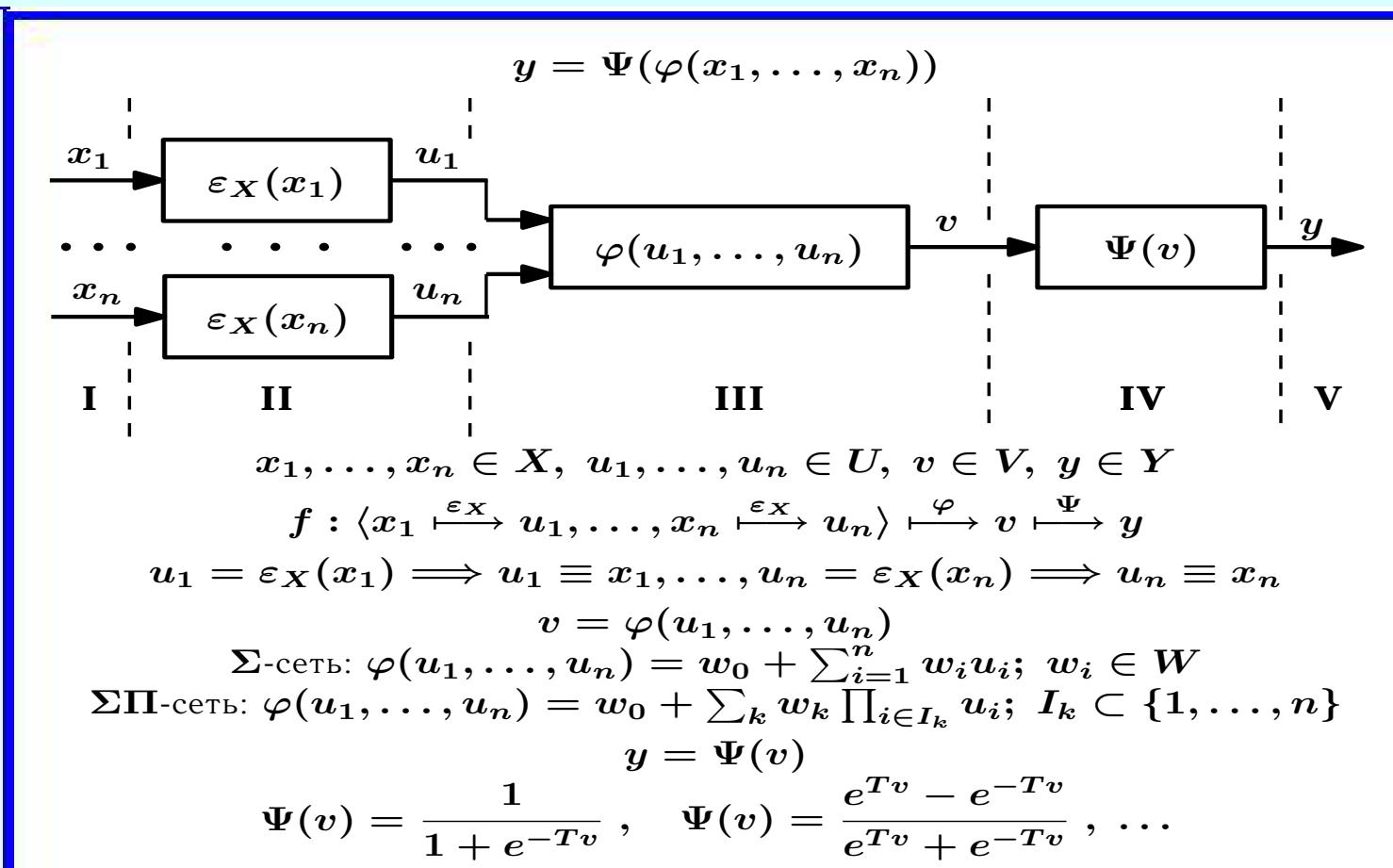
Результат:

$$y$$

Нейросети и составное представление функций (ХХIII)

Сопоставление нейросетевого подхода с «классикой» – 8

Структура модуля нейронной сети прямого распространения



Нейросети и составное представление функций (XXIV)

Сопоставление нейросетевого подхода с «классикой» – 9

Пример 2. Разложение по системе функций (на примере интерполяционного полинома Лагранжа)

Исходные данные:

$$x$$

Входные отображения:

$$f_i(x) \Rightarrow a_i x^{i-1} = u_i$$

Сжимающее отображение:

$$\varphi(u_1, \dots, u_n) \Rightarrow \sum_{i=0}^n u_i = v$$

Выходное отображение:

$$\Psi(v) \Rightarrow v = y$$

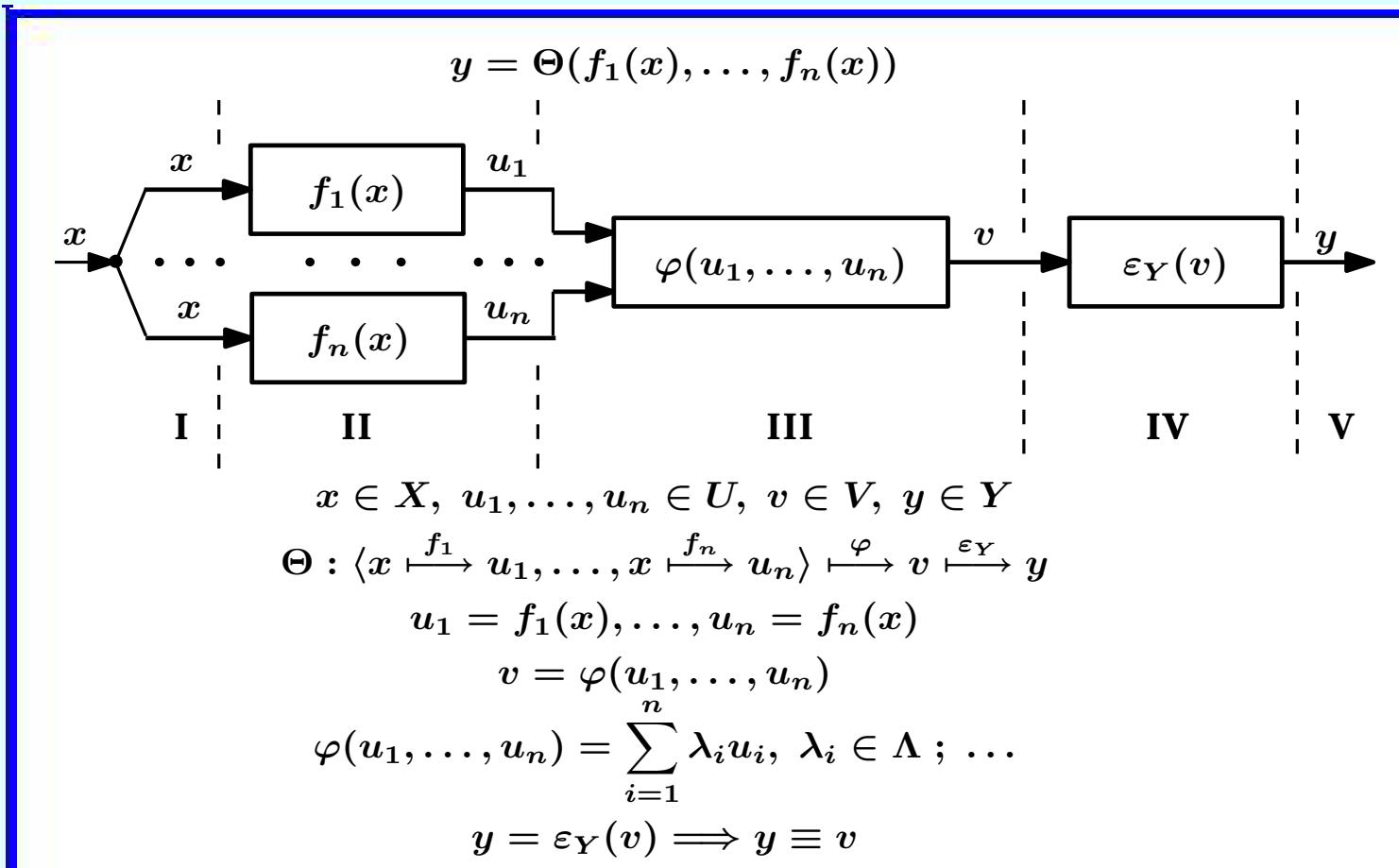
Результат:

$$y$$

Нейросети и составное представление функций (XXV)

Сопоставление нейросетевого подхода с «классикой» – 10

Конкретизация базисного модуля композиционной модели –
функциональное разложение



Нейросети и составное представление функций (XXVI)

Сопоставление нейросетевого подхода с «классикой» – 11

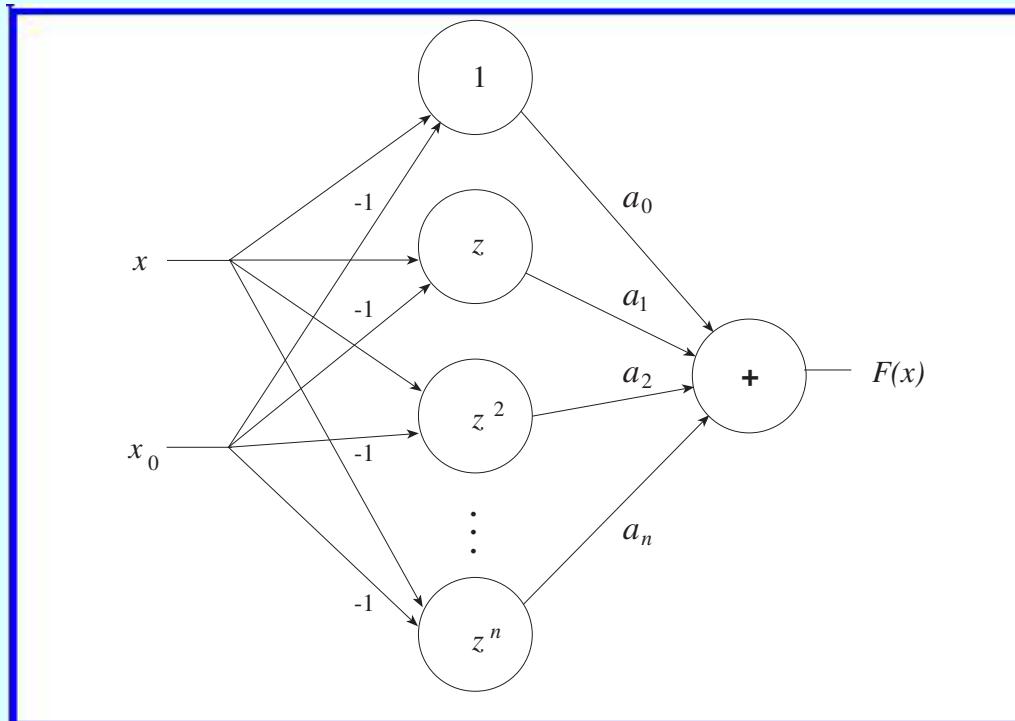
	Элемент КМ	Нейросеть	Полином Лагранжа
ИД	x	(x_1, \dots, x_n)	x
ВхО	$f_i(x_i)$	$x_i = u_i$	$a_i x^{i-1} = u_i$
СжО	$\varphi(u_1, \dots, u_n)$	$\sum_{i=0}^n w_i u_i = v$	$\sum_{i=0}^n u_i = v$
ВыхО	$\Psi(v)$	$\frac{1}{1+e^{-Tv}} = y$	$v = y$
Рез	y	y	y

Здесь: **ИД** – исходные данные; **ВхО** – входные отображения;
СжО – сжимающее отображение; **ВыхО** – выходное
отображение; **Рез** – результат.

Нейросети и составное представление функций (XXVII)

Сопоставление нейросетевого подхода с «классикой» – 12

Сеть Тейлора



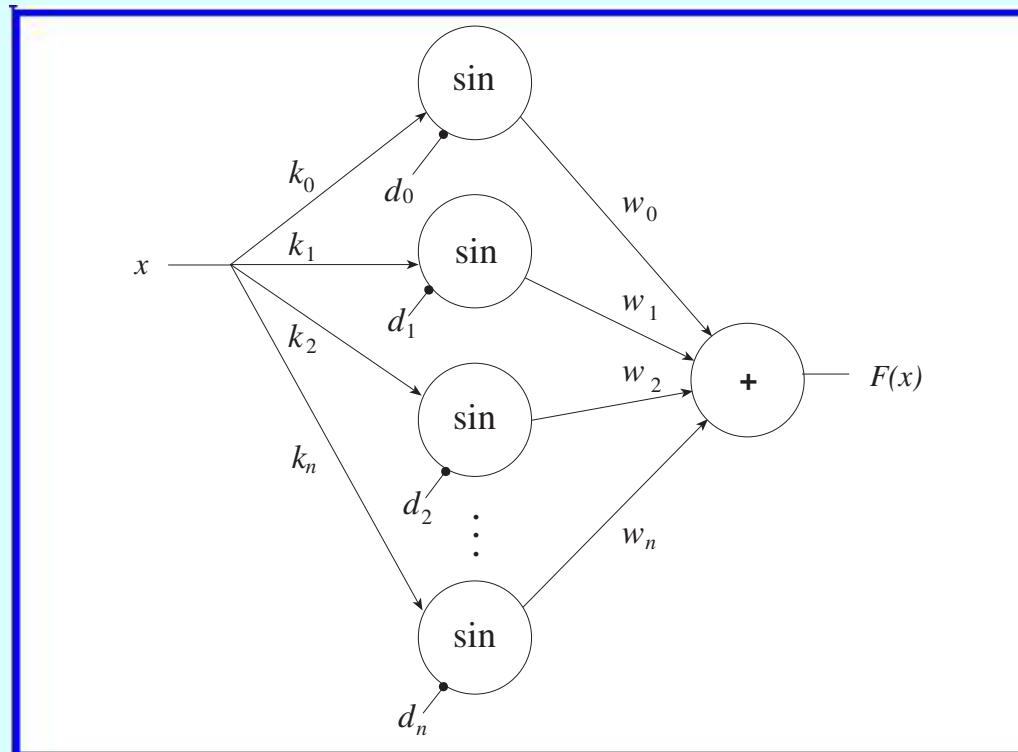
$$F(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \cdots + a_n(x - x_0)^n + \cdots$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 1.16, p. 25)

Нейросети и составное представление функций (XXVIII)

Сопоставление нейросетевого подхода с «классикой» – 13

Сеть Фурье



$$F(x) = \sum_{i=0}^{\infty} (a_i \cos(ix) + b_i \sin(ix)), \quad \sin(x + d_0) = \cos(x), \text{ если } d_0 = \pi/2$$

Источник: Rojas R. Neural networks: A systematic introduction. – Springer, 1996. – 453 pp.
(Figure 1.17, p. 25)

Нейросети и составное представление функций (XXIX)

Сопоставление нейросетевого подхода с «классикой» – 14

Основные выводы:

1. Наглядно видна **специфика нейронных сетей**, которая состоит в следующем:

- в «конструкции» **сжимающего отображения** (эта черта **не слишком принципиальна**);
- в методах подбора параметров (синаптических весов, чаще всего) НС-моделей (**«обучение»**) — **умеренно принципиальная** черта;
- аналогом «классического» разложения является фактически единственный искусственный нейрон (не сеть в целом!); базисный модуль, из которых состоит КМ «нейросетевого» типа, **обязательно** содержит **композицию** сжимающего отображения с выходным отображением (активационной функцией), это **принципиально** отличает КМ «нейросетевого» типа от КМ «классического» типа;
- «однонейронные» модели используются крайне редко, типичные нейросетевые модели представляют собой сети из связанных между собой нейронов; это обстоятельство, особенно в слоистых сетях, резко повышает **число «степеней свободы»** НС-модели в сравнении с моделью «классической», определяемое количеством параметров сети, которые можно варьировать, что обеспечивает чрезвычайно высокую гибкость НС-моделей; **сетевое объединение (многоуровневая композиция)** базисных модулей КМ (нейронов) — это **очень принципиально**, аналогов этому в традиционных моделях нет.

Нейросети и составное представление функций (XXX)

Сопоставление нейросетевого подхода с «классикой» – 15

Основные выводы:

2. В то же время, НС-модели **не обладают** какой-либо **«исключительностью»**:

НС-модели представляют собой «всего лишь» **один из видов композиционных моделей**, возможны и другие варианты КМ, в частности, те, которые реализуют «классические» модели численного моделирования.

Нейросети и составное представление функций (XXXI)

Пределы возможностей нейросетевого подхода – 1

Рассмотрим теперь **третий из вопросов**, сформулированных ранее:

- 1) Неужели ничего **похожего** на НС-представления не было?
- 2) В чем же состоят **отличия** НС-варианта от «классики»?
- 3) В чем **сила** НС-подхода и каковы **пределы его возможностей**?

Нейросети и составное представление функций (XXXII)

Пределы возможностей нейросетевого подхода – 2

Теорема Колмогорова (1)

Проблемы Гильберта — список из **23** кардинальных проблем математики, представленный **Давидом Гильбертом** на II Международном Конгрессе математиков в Париже в **1900** году.

При решении **13-й проблемы** была сформулирована достаточно **общая задача**, формулировка которой звучит следующим образом:

Можно ли произвольную непрерывную функцию **n** переменных получить с помощью операций сложения, умножения и суперпозиции из непрерывных функций **двух** переменных?

Ответ оказался **положительным**. В серии работ **А. Н. Колмогоров** и **В. И. Арнольд** решили эту проблему в постановке даже еще более жесткой и получили следующий **результат**:

Любую непрерывную функцию **n** переменных можно получить с помощью операций сложения, умножения и суперпозиции из непрерывных функций **одной** переменной.

Нейросети и составное представление функций (XXXIII)

Пределы возможностей нейросетевого подхода – 3

Теорема Колмогорова (2)

Формулировка теоремы Колмогорова

Каждая непрерывная функция n переменных, заданная на единичном кубе n -мерного пространства, представима в виде

$$f(x_1, x_2, \dots, x_n) = \sum_{q=1}^{2n+1} h_q \left[\sum_{p=1}^q \varphi_q^p(x_p) \right]$$

где функции $h_q(u)$ непрерывны, а $\varphi_q^p(x_p)$, кроме того, еще и стандартны, т. е. не зависят от выбора функции f .

Существенно то, что теорема Колмогорова говорит о **точном представлении** функций многих переменных с помощью функции одной переменной.

Нейросети и составное представление функций (XXXIV)

Пределы возможностей нейросетевого подхода – 4

Теорема Вейерштрасса-Стоуна-Горбаня (1)

Теорема Колмогорова относится к **точному представлению** функций. Кроме вопроса о точном представлении, существует еще один — **об аппроксимации**, существенно более важный для прикладных задач.

По знаменитой **теореме Вейерштрасса** непрерывную функцию нескольких переменных $f(x_1, x_2, \dots, x_n)$ на замкнутом ограниченном множестве Q можно равномерно приблизить последовательностью полиномов.

Теорема Стоуна представляет собой обобщение теоремы Вейерштрасса, расширяющее класс рассматриваемых функций.

Нейросети и составное представление функций (XXXV)

Пределы возможностей нейросетевого подхода – 5

Теорема Вейерштрасса-Стоуна-Горбаня (2)

А. Н. Горбань (1998) обобщил результат Стоуна. Теоремы, полученные им, можно трактовать как утверждения об **универсальных аппроксимационных свойствах любой нелинейности**: с помощью линейных операций и каскадного соединения можно из произвольных нелинейных элементов получить аппроксимацию **любого отображения**

$$\Psi : X^n \rightarrow Y^m$$

с любой наперед заданной точностью.

Горбань А.Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей // Сибирский журнал Вычислительной математики. – 1998. – том 1, № 1. – с. 11–24.

Горбань А.Н. Обобщенная аппроксимационная теорема и точное представление многочленов от нескольких переменных суперпозициями многочленов от одного переменного // Известия высших учебных заведений. Математика. – 1998. – № 5 (432) . – с. 6–9.

Горбань А.Н. Функции многих переменных и нейронные сети // Соросовский образовательный журнал. – 1998. – № 12 . – с. 105–112.

Модели с обратными связями (I)

Рекуррентные сети – 1

Сети без обратных связей (сети прямого распространения):

- сети, **обучаемые с учителем**, задающим образцы правильных ответов;
- сети, **обучаемые без учителя**, которые адаптируют свою структуру к данным, не требуя сведений об их принадлежности к тому или иному классу.

Характерные особенности сетей прямого распространения:

- обученная сеть выдает ответ **сразу после прохождения** через нее входного сигнала;
- каждый нейрон при прохождении входного сигнала через сеть **срабатывает лишь однажды**;
- сложная многоэтапная обработка информации требует наличия в сети **нескольких слоев**.

Модели с обратными связями (II)

Рекуррентные сети – 2

Рекуррентные сети – особенности:

- естественное обобщение однопроходных сетей прямого распространения;
- новое качество рекуррентных сетей – динамическая обработка информации:
 - выходы сети **возвращаются обратно** на входы;
 - информация пропускается через сеть **многократно**.

Аналогия:

сеть прямого распространения (<i>«статика»</i>)	VS	рекуррентная сеть (<i>«динамика»</i>)
↓ «обычная» формула $y = f(x)$	VS	рекуррентная формула $w_{n+1} = w_n + \Delta w_n$

Модели с обратными связями (III)

Рекуррентные сети – 3

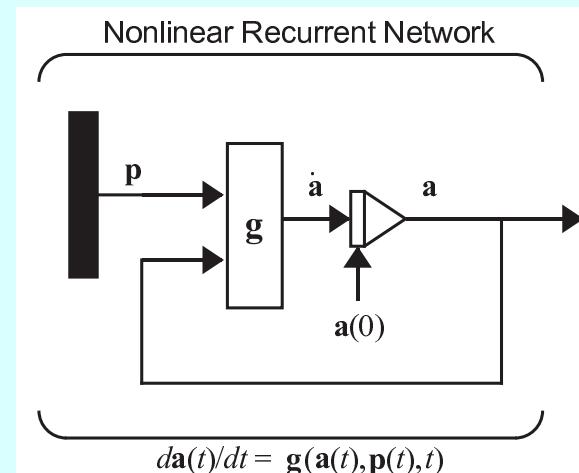
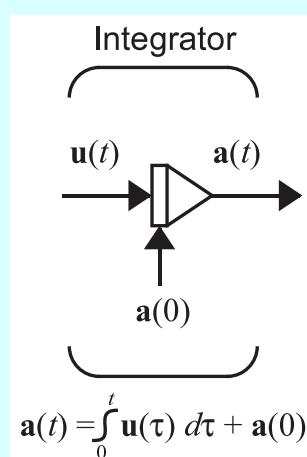
Рекуррентные сети — достоинства и недостатки:

- ❑ существенно **более высокие возможности** моделирования систем различных видов, включая динамические системы;
- ❑ существенно **более сложные** процессы обучения;
- ❑ возможные **проблемы устойчивости** процессов функционирования обученных сетей.

Модели с обратными связями (IV)

Рекуррентные сети – 4

Нелинейная рекуррентная сеть с непрерывным временем



Интегратор:

$$a(t) = \int_0^t u(\tau) d\tau + a(0)$$

Нелинейная рекуррентная сеть:

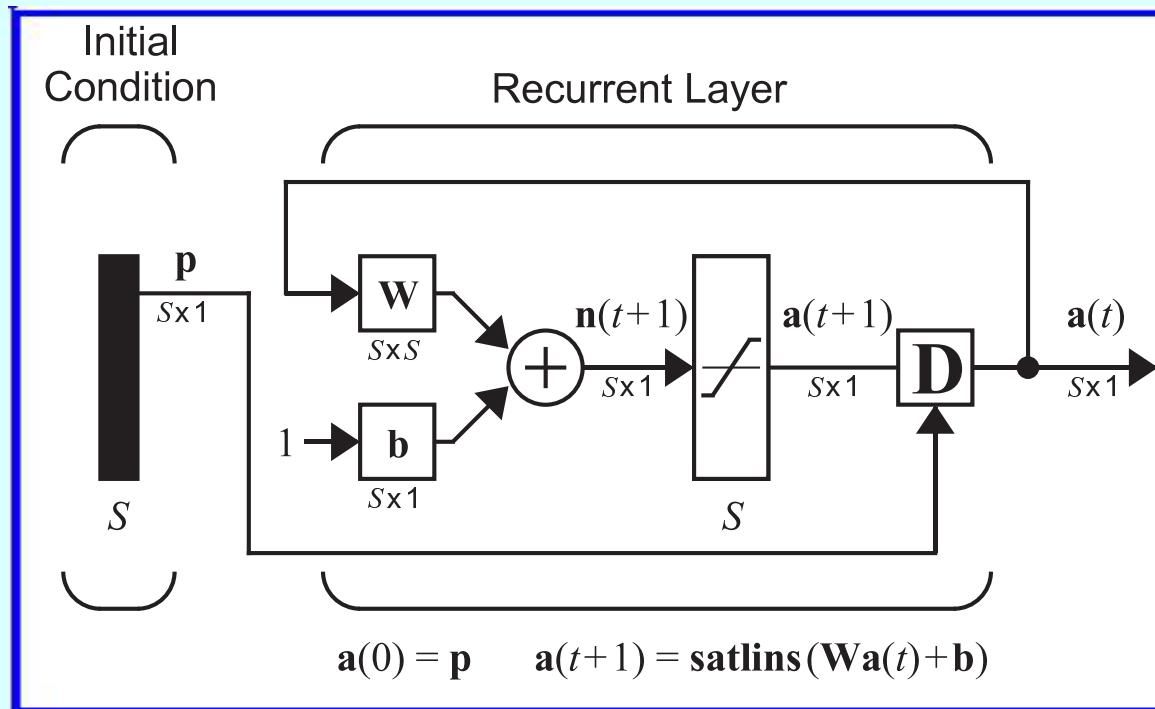
$$\frac{d}{dt} a(t) = g(a(t), p(t), t)$$

Вопрос: При каких условиях процесс, реализуемый данной нелинейной динамической системой, сходится к некоторому **устойчивому** состоянию?

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 2, Figure 2.12, p.2-14; Chapter 17, Figure 17.1, p.17-2).

Модели с обратными связями (V)

Сеть Хопфилда

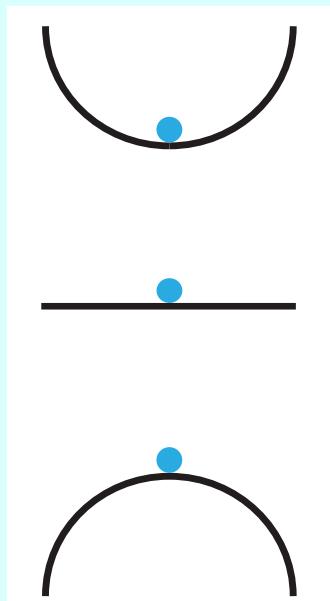


Проблема: Требуется так сформировать и настроить сеть Хопфилда, чтобы паттерны-эталоны, запомненные ею, были **аттракторами** и обладали большими **областями притяжения**.

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Figure 3.6, p.3-12).

Модели с обратными связями (VI)

Понятие устойчивости



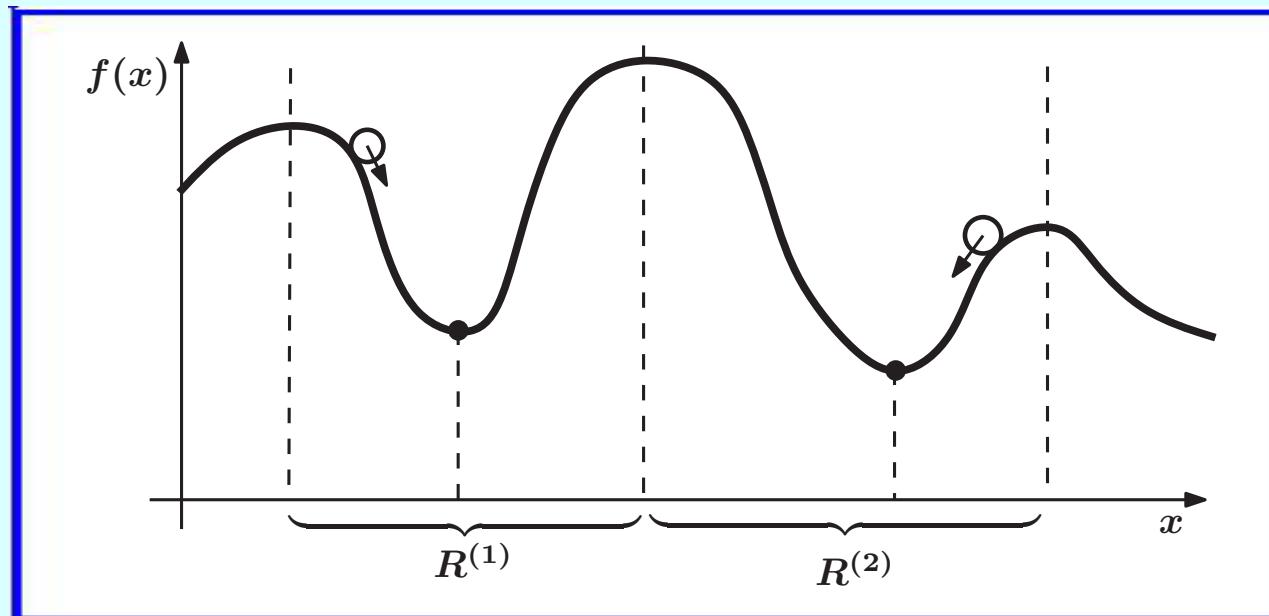
Виды систем:

- устойчивая** система (верхний рисунок);
- нейтральная** система (средний рисунок);
- неустойчивая** система (нижний рисунок).

Источник: *Hagan M. T., Demuth H.B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 17, p.17-3).

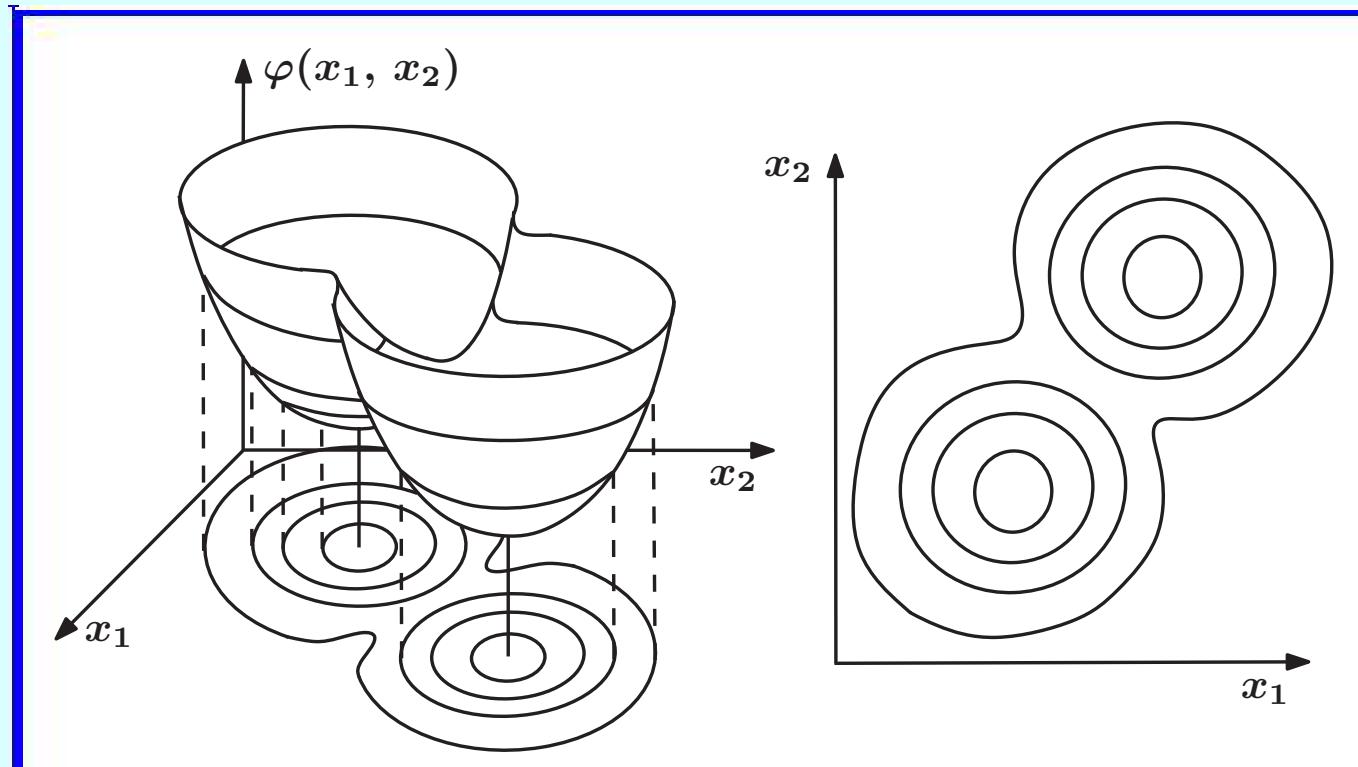
Модели с обратными связями (VII)

Понятие области притяжения — одномерный случай



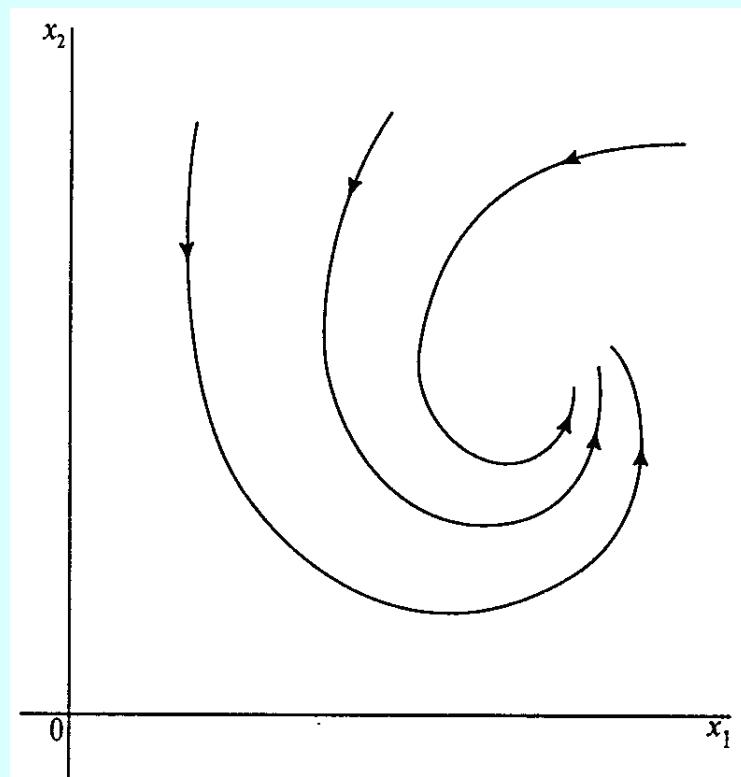
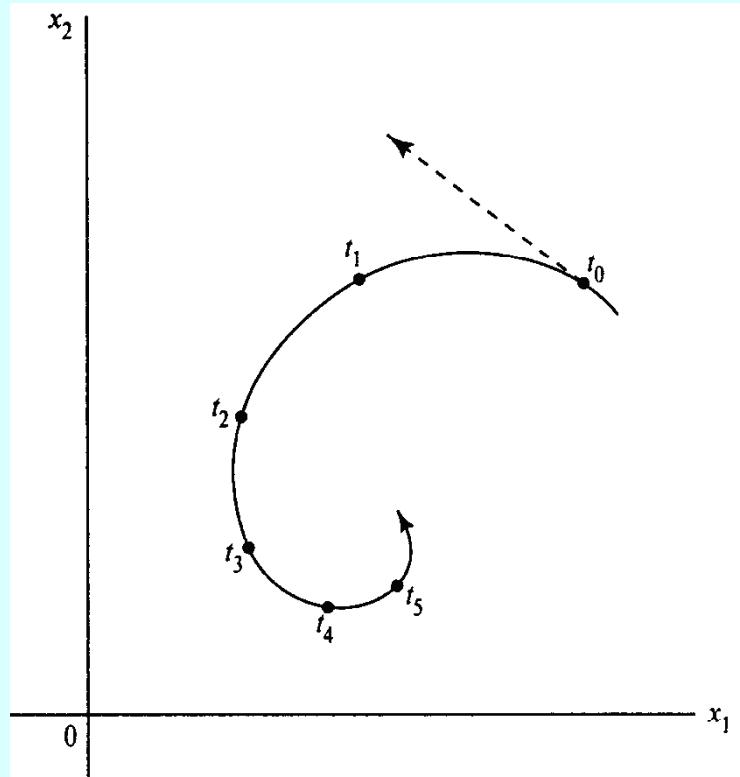
Модели с обратными связями (VIII)

Понятие области притяжения — двумерный случай



Модели с обратными связями (IX)

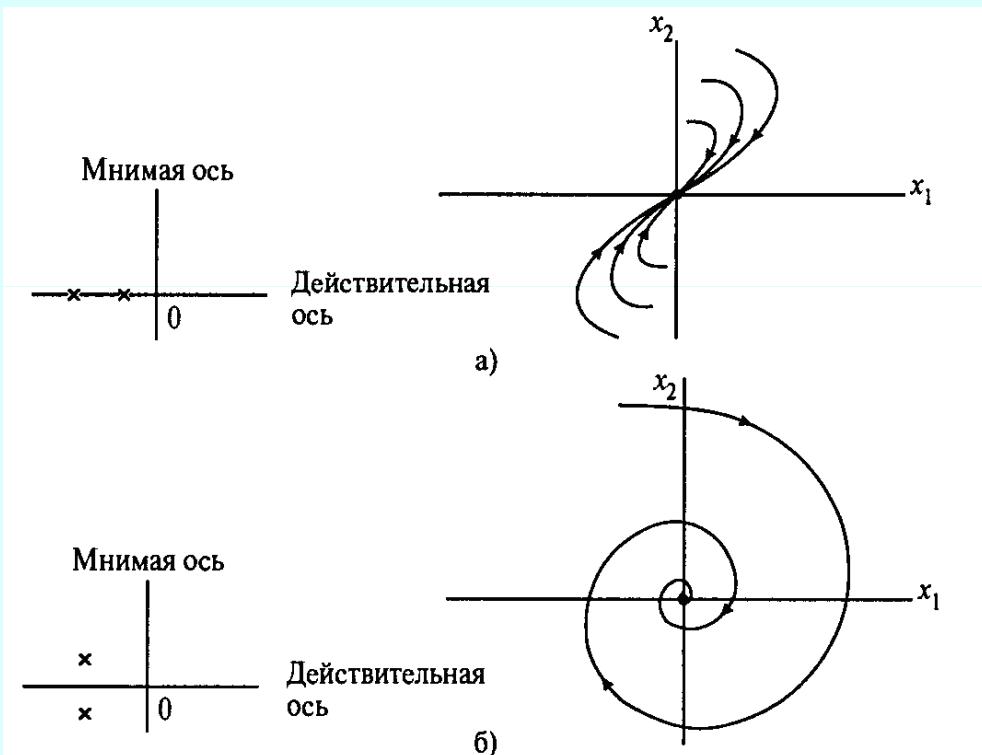
Фазовый портрет динамической системы



Источник: Хайкин С. Нейронные сети: Полный курс. – 2-изд. – М.: Вильямс, 2006. – 1104 с.
(Гл.14, рис.14.1, 14.2, с.839–840).

Модели с обратными связями (Х)

Фазовый портрет динамической системы



Виды состояний равновесия

(1)

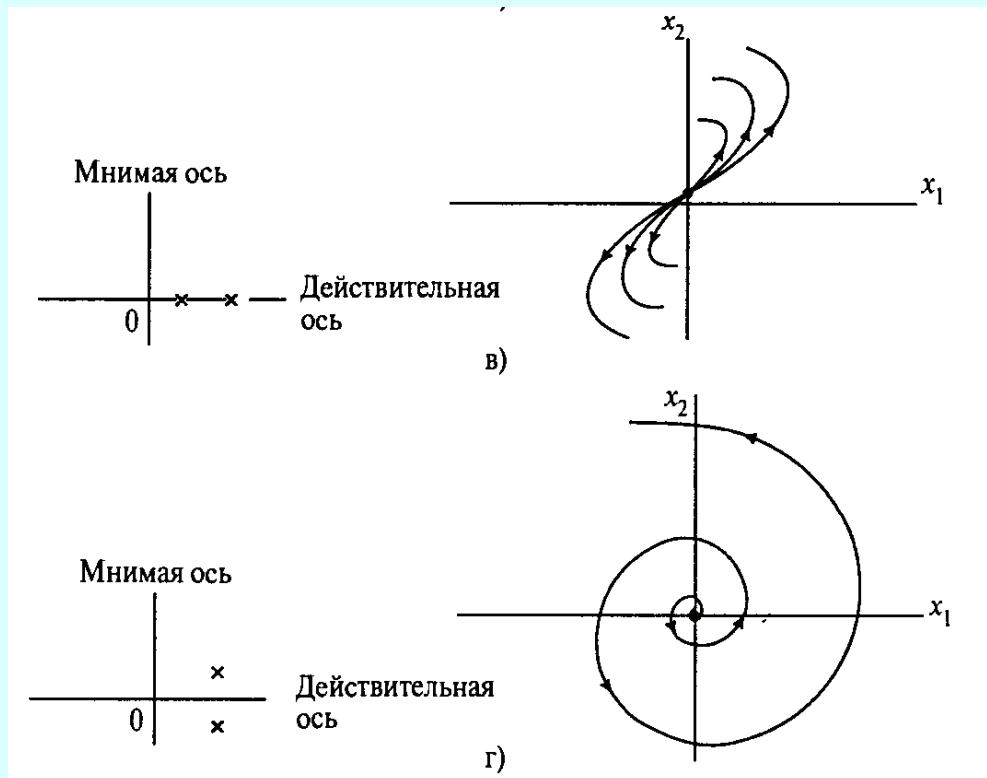
- устойчивый узел (а);
- устойчивый фокус (б).

Все фазовые траектории **входят в начало координат $x = 0$** , которое в данном случае является **состоянием устойчивого равновесия** динамической системы (система устойчива).

Источник: Хайкин С. Нейронные сети: Полный курс. – 2-изд. – М.: Вильямс, 2006. – 1104 с.
(Гл.14, рис.14.4, с.844).

Модели с обратными связями (XI)

Фазовый портрет динамической системы



Виды состояний равновесия (2)

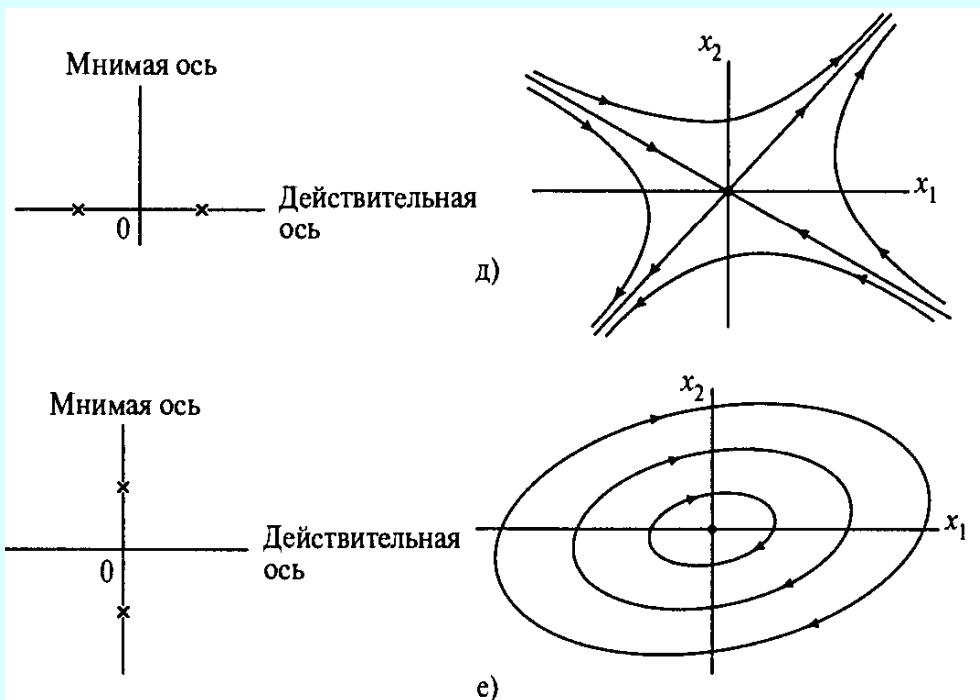
- неустойчивый узел (в);
- неустойчивый фокус (г).

Все фазовые траектории **исходят из начала координат $x = 0$** . В данном случае динамическая система **не имеет** состояния устойчивого равновесия (система неустойчива).

Источник: Хайкин С. Нейронные сети: Полный курс. – 2-изд. – М.: Вильямс, 2006. – 1104 с.
(Гл.14, рис.14.4, с.844).

Модели с обратными связями (XII)

Фазовый портрет динамической системы



Виды состояний равновесия (3)

- седловая точка (д);
- центр (е).

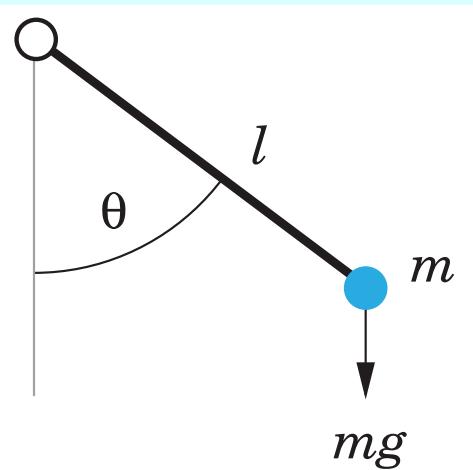
Случай (д): Все фазовые траектории, **входящие** в начало координат, устойчивы, траектории, **исходящие** из начала координат – неустойчивы.

Случай (е): Траектории **никогда** не приходят в начало координат и не исходят из него, динамическая система реализует **пределочный цикл**.

Источник: Хайн С. Нейронные сети: Полный курс. – 2-изд. – М.: Вильямс, 2006. – 1104 с.
(Гл.14, рис.14.4, с.844).

Модели с обратными связями (XIII)

Пример динамической системы – 1



Маятник (1)

Уравнение движения маятника:

$$ml \frac{d^2\theta}{dt^2} + c \frac{d\theta}{dt} + mg \sin \theta = 0$$

m — масса маятника

c — коэффициент демпфирования

Положения устойчивого равновесия:

$$\theta = 2\pi n, n = 0, \pm 1, \pm 2, \dots$$

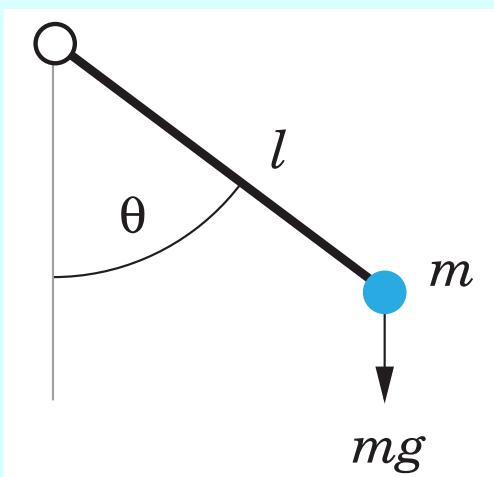
Положения неустойчивого равновесия:

$$\theta = \pi n, n = 1, 3, 5, \dots$$

Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 17, Figure 17.3, p.17-6).

Модели с обратными связями (XIV)

Пример динамической системы – 2



Маятник (2)

Уравнение движения маятника
в переменных состояния:

$$a_1 = \theta, \quad a_2 = \frac{d\theta}{dt}$$

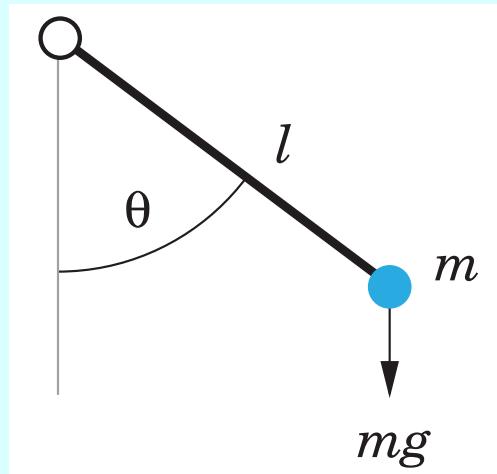
$$\frac{da_1}{dt} = a_2$$

$$\frac{da_2}{dt} = -\frac{g}{l} \sin a_1 - \frac{c}{ml} a_2$$

Источник: *Hagan M. T., Demuth H.B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 17, Figure 17.3, p.17-6).

Модели с обратными связями (XV)

Пример динамической системы – 3



Маятник (3)

Полная энергия маятника:

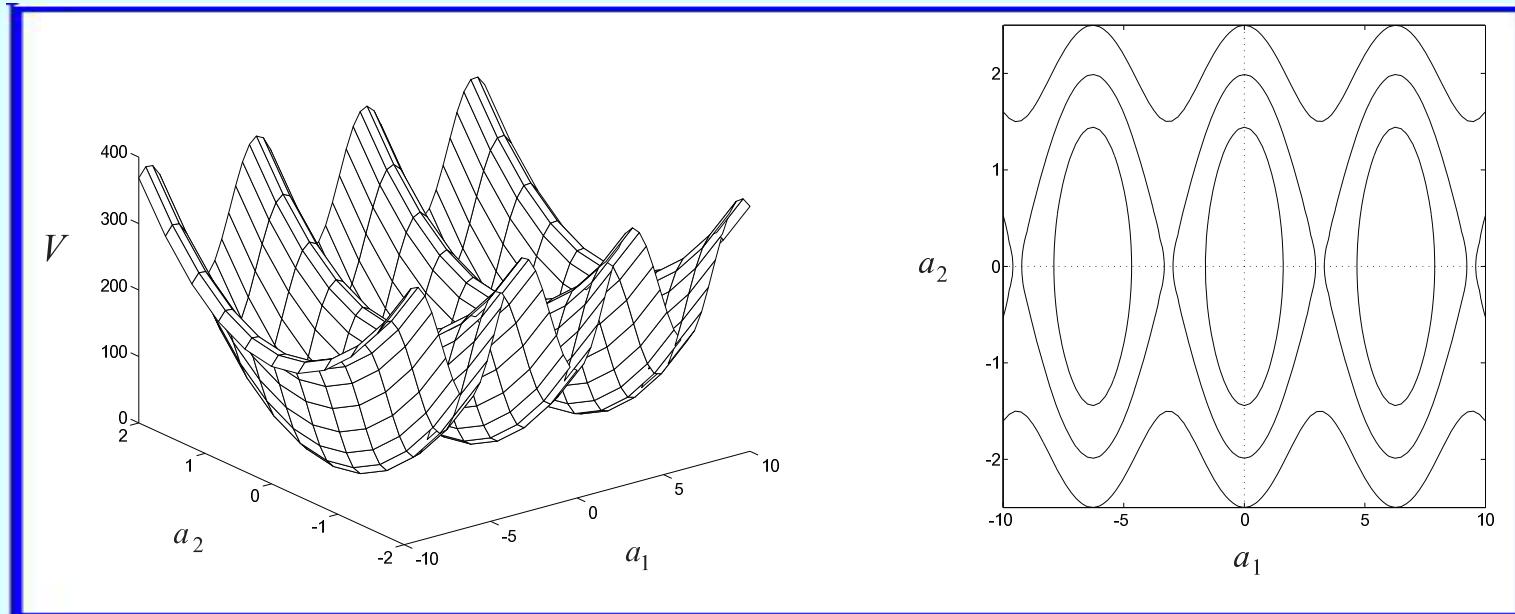
$$V(a) = \underbrace{\frac{1}{2}ml^2a_2^2}_{\text{Кинетическая энергия}} +$$

$$+ \underbrace{mgl(1 - \cos a_1)}_{\text{Потенциальная энергия}}$$

Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 17, Figure 17.3, p.17-6).

Модели с обратными связями (XVI)

Пример динамической системы – 4



$$g = 9.8, \quad m = 1, \quad l = 9.8, \quad c = 1.96$$

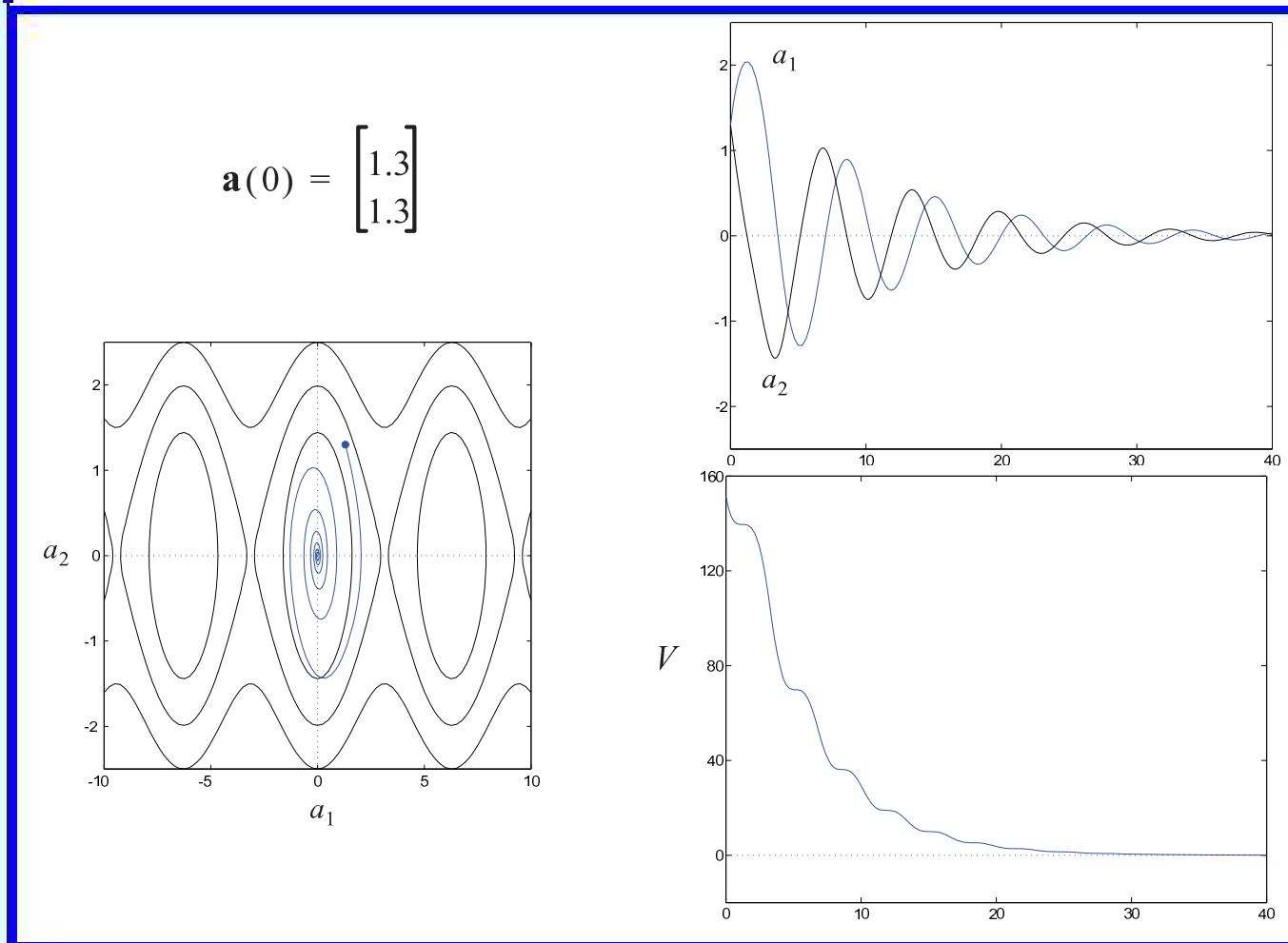
$$(-10 \leq \theta \leq +10) \text{ rad}, \quad (-2 \leq \dot{\theta} \leq +2) \text{ rad/sec}$$

$-2\pi, 0, +2\pi$ – **точки минимума** функции энергии

Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 17, Figure 17.4, p.17-9).

Модели с обратными связями (XVII)

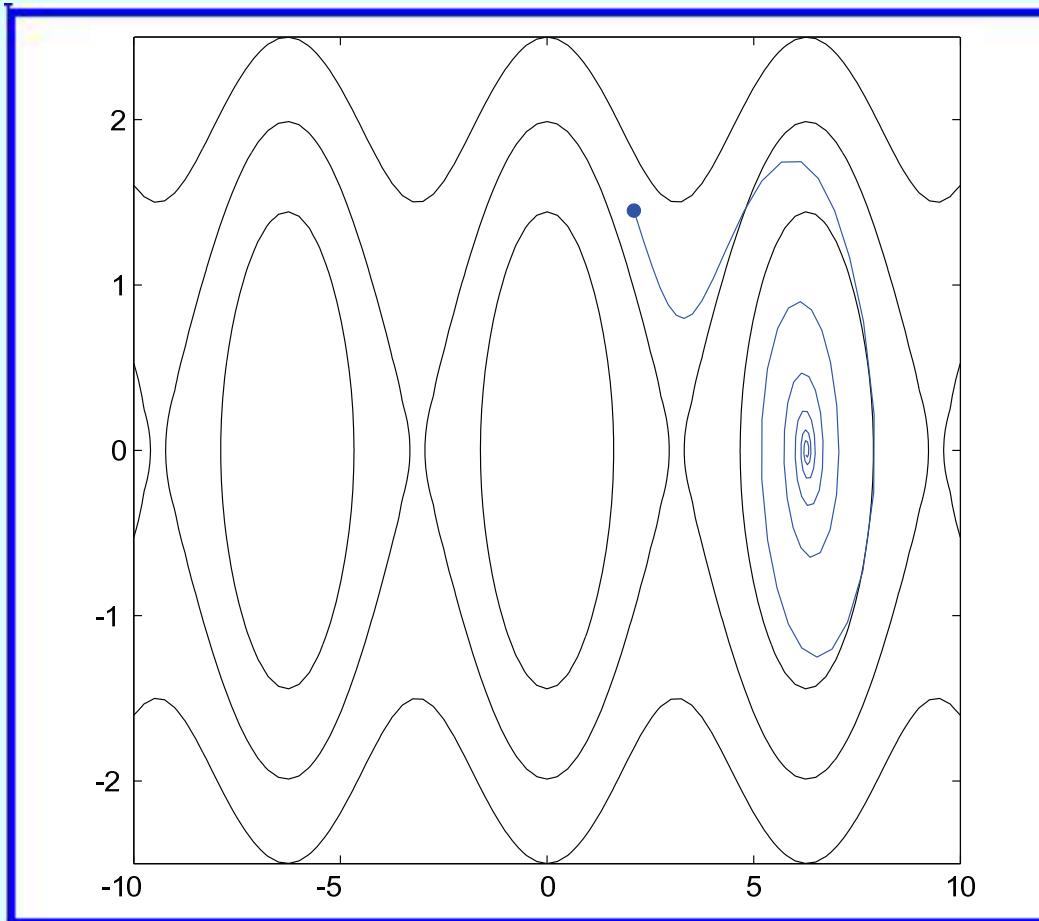
Пример динамической системы – 5



Источник: Hagan M. T., Demuth H. B., Beale M. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 17, Figures 17.5, 17.6, 17.7, p.17-10, 17-11).

Модели с обратными связями (XVIII)

Пример динамической системы – 6



$$a_1(0) = \theta(0) = 2 \text{ rad}, \quad a_2(0) = \dot{\theta}(0) = 1.5 \text{ rad/sec}$$

Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 17, Figure 17.13, p.17-18).

Модели с обратными связями (XIX)

Сети Хопфилда — истоки

Джон Хопфилд (John Hopfield) (1982, 1984),
полносвязная рекуррентная сеть, основанная на аналогиях с моделями
статистической физики (спиновые стекла):

Hopfield J. Neural networks and physical systems with emergent collective abilities // *Proc. National Academy of Sciences, USA.* – April **1982**. – Vol. 79. – pp. 2554–2558.

Hopfield J. Neural networks with graded response have collective computational properties like those of two-state neurons // *Proc. National Academy of Sciences, USA.* – May **1984**. – Vol. 81. – pp. 3088–3092.

В работе Хопфилда впервые было обращено внимание на аналогию между **сетями с симметричными связями** и давно известными физикам объектами — **спиновыми стеклами**, которые исследовались с помощью методов статистической механики.

Было показано, что такие сети могут послужить основой для построения моделей **ассоциативной памяти**.

С этих работ Хопфилда началось **возрождение и быстрое развитие** тематики нейросетевого моделирования, которое существенно затормозилось после выхода книги Минского и Пейпера.

Модели с обратными связями (ХХ)

Физическая основа сетей Хопфилда – 1



Знаки связей между спинами в ферромагнетике, антиферромагнетике и спиновом стекле

Взаимодействие друг с другом атомов, обладающих магнитными моментами, в кристаллической решетке:

Ферромагнетики — в состоянии минимальной энергии все атомы в кристаллической решетке ориентируют свои моменты параллельно друг другу; связи между атомами описываются одинаковыми **положительными** числами.

Антиферромагнетики — все связи **отрицательны**, соседние спины ориентируются в **противоположных** направлениях.

Спиновые стекла — связи между магнитными моментами атомов имеют **случайный характер**.

Источник: Ежов А.А., Шумский С.А. Нейрокомпьютинг и его приложения в экономике и бизнесе. – М.: Изд-во МИФИ, 1998. – 224 с. (рис.1, с.90)

Модели с обратными связями (XXI)

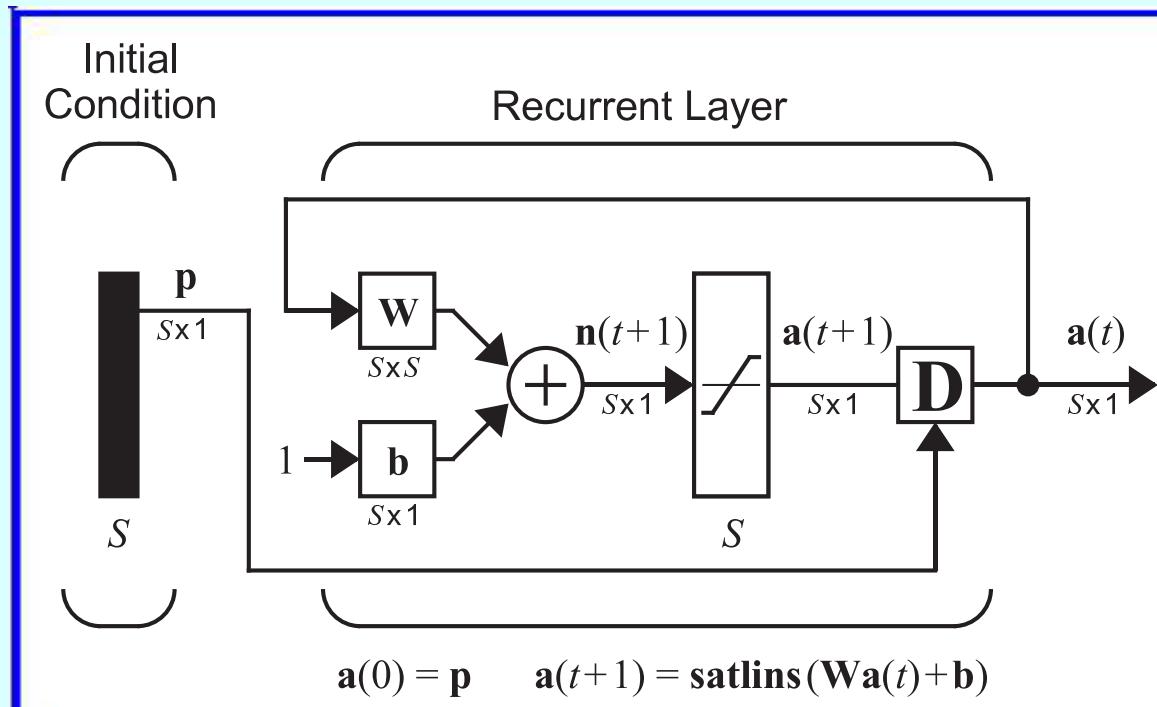
Физическая основа сетей Хопфилда – 2

Сеть Хопфилда — **полносвязная** рекуррентная нейронная сеть с **симметричными** связями между нейронами.

Сети Хопфилда, подобно **спиновым стеклам**, имеют множество **стационарных конфигураций** активности нейронов, являющихся **аттракторами**, т.е. такими состояниями, к которым сходится динамика сети.

Модели с обратными связями (ХII)

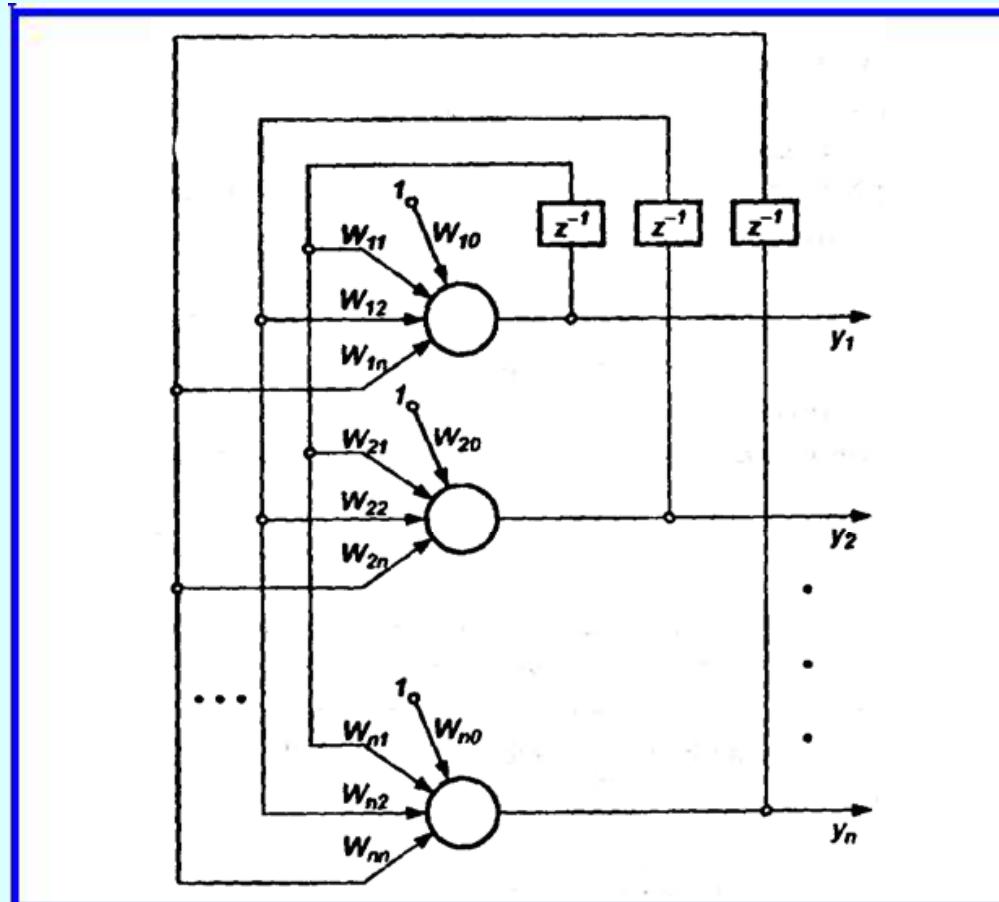
Структура сети Хопфилда – 1



Источник: *Hagan M. T., Demuth H.B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 3, Figure 3.6, p.3-12).

Модели с обратными связями (ХХIII)

Структура сети Хопфилда – 2



Источник: Осовский С. Нейронные сети для обработки информации: Пер. с польск. – М.: Финансы и статистика, 2002. – 344 с. (Глава 7, рис.7.1, с.178).

Модели с обратными связями (XXIV)

Сеть Хопфилда как ассоциативная память – 1

Симметричность связей

Матрица связей \mathbf{W} сети Хопфилда:

- полная и симметричная ($w_{ij} = w_{ji}$);
- самовоздействие нейронов отсутствует ($w_{ii} = 0$).

Многослойные сети: входные и выходные нейроны пространственно разделены.

Сеть Хопфилда: все нейроны одновременно являются и входными, и скрытыми, и выходными.

Входы-выходы сети Хопфилда: *роль входа* выполняет начальная конфигурация активностей нейронов, а *роль выхода* — конечная стационарная конфигурация их активностей.

Модели с обратными связями (XXV)

Сеть Хопфилда как ассоциативная память – 2

Асинхронная и синхронная динамика (1)

Нейроны в модели Хопфилда, подобно спиновым переменным, могут принимать **два состояния**: $s_i \in \{-1, +1\}$, а динамика сети носит **асинхронный характер**.

В дискретные моменты времени $t = 1, 2, \dots$ **случайным образом** выбирается **один нейрон** (с номером k), для которого вычисляется значение **потенциала**:

$$h_k = \sum_j w_{kj} s_j$$

Если выполняется **условие** $h_k s_k < 0$, то состояние k -го нейрона изменяется на **противоположное**:

$$s_k \rightarrow -s_k$$

Последовательный вариант асинхронной динамики: перебор нейронов производится **не случайным образом**, а **циклически**, но в каждый момент времени изменяется состояние **только одного** нейрона.

Модели с обратными связями (XXVI)

Сеть Хопфилда как ассоциативная память – 3

Асинхронная и синхронная динамика (2)

Качественно другой подход – **синхронная динамика** (параллельная динамика).

В синхронной динамике **одновременно изменяются** состояния всех тех нейронов, для которых выполняется условие $h_k s_k < 0$.

Синхронизация моментов обновления состояний нейронов делает такую динамику **подверженной «зацикливаниям»**.

Модели с обратными связями (XXVII)

Сеть Хопфилда как ассоциативная память – 4

Метрика пространства состояний (1)

Расстояние между состояниями сети
задается в метрике Хемминга:

Хеммингово расстояние между бинарными векторами $b^{(1)}$ и $b^{(2)}$ определяется как количество различающихся компонент в них:

$$\rho_H(b^{(1)}, b^{(2)}) = \sum_i (b_i^{(1)} - b_i^{(2)})^2$$

Пример:

$$b^{(1)} = (1, 0, 0, 0, 1), \quad b^{(2)} = (1, 1, 0, 0, 0), \quad \rho_H(b^{(1)}, b^{(2)}) = 2$$

Модели с обратными связями (XXVIII)

Сеть Хопфилда как ассоциативная память – 5

Метрика пространства состояний (2)

В случае **спиновых переменных** $s_i^{(1)}, s_i^{(2)}$, принимающих значения ± 1 ,
расстояние Хемминга принимает вид:

$$\rho_H(s^{(1)}, s^{(2)}) = \frac{1}{2} \left(N - \sum_i s_i^{(1)} s_i^{(2)} \right) = \frac{1}{2} (N - s^{(1)} s^{(2)})$$

Здесь $s^{(1)} s^{(2)}$ – скалярное произведение векторов $s^{(1)}$ и $s^{(2)}$.

Модели с обратными связями (XXIX)

Сеть Хопфилда как ассоциативная память – 6

Энергия состояния (1)

Асинхронная динамика сети Хопфилда сопровождается уменьшением энергии сети, определяемой для случая нулевых порогов активации (смещений) как:

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j$$

Если j -й элемент изменяет свое состояние на Δs_j , изменение энергии равно

$$\Delta E = -\Delta s_j \sum_i w_{ij} s_i$$

Изменение энергии сети – функция $\Delta E(\Delta s_j, \sum_i w_{ij} s_i)$:

Старое s_j	$\sum_i w_{ij} s_i$	Новое s_j	Δs_j	ΔE
> 0	> 0	> 0	> 0	< 0
> 0	< 0	< 0	< 0	< 0
< 0	< 0	< 0	< 0	< 0
< 0	> 0	> 0	> 0	< 0

Δs_j и $\sum_i w_{ij} s_i$ всегда имеют одинаковый знак, поэтому энергия сети при переходе от итерации к итерации всегда уменьшается.

Модели с обратными связями (XXX)

Сеть Хопфилда как ассоциативная память – 7

Энергия состояния (2)

Для случая **ненулевых порогов** активации (смещений) θ_i **энергия сети** определяется следующим образом:

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

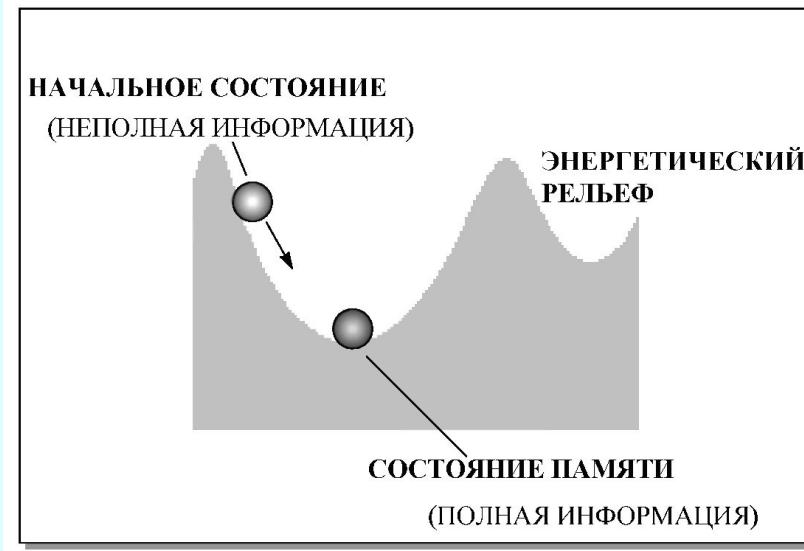
Поскольку число нейронов сети **конечно**, функция энергии **ограничена снизу**.

Следовательно, эволюция состояния сети должна закончиться в **стационарном состоянии**, которому будет соответствовать **локальный минимум энергии**.

Модели с обратными связями (XXXI)

Сеть Хопфилда как ассоциативная память – 8

Энергия состояния (3)



Изменение состояния сети Хопфилда

Эволюция состояния сети заканчивается в **стационарном состоянии**, соответствующий локальному **минимуму функции энергии**.

Локальные минимумы функции энергии в сети Хопфилда являются **устойчивыми состояниями памяти**, а окружающие точки на рельефе этой функции — **переходными состояниями**.

Такая динамика определяет главное свойство сети Хопфилда — **способность восстанавливать состояние равновесия** из некоторого начального возмущенного состояния, т.е. **«вспоминать»** искаженные или потерянные биты информации.

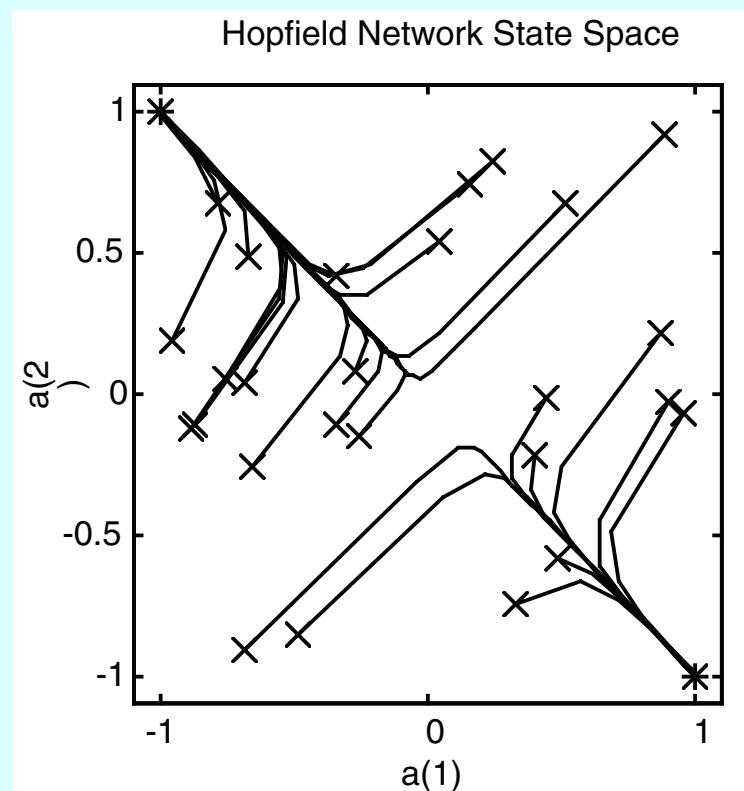
Восстановление полной информации по какой-либо ее части (вспоминание по ассоциации) наделяет модель Хопфилда свойством **ассоциативной памяти**.

Источник: Ежов А.А., Шумский С.А. Нейрокомпьютинг и его приложения в экономике и бизнесе. – М.: Изд-во МИФИ, 1998. – 224 с. (рис.4, с.92)

Модели с обратными связями (XXXII)

Сеть Хопфилда как ассоциативная память – 8

Фазовые траектории – пример



Изменение состояний сети Хопфилда

Запоминаемые векторы:

$$\sigma^{(1)} = (+1 \quad -1)$$

$$\sigma^{(2)} = (-1 \quad +1)$$

Весовая матрица и смещения:

$$W = \begin{bmatrix} 0.6925 & -0.4694 \\ -0.4694 & 0.6925 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 13-14).

Модели с обратными связями (XXXIII)

Сеть Хопфилда как ассоциативная память – 10

Ассоциативная память (4)

Аттракторы сети Хопфилда — ее **стационарные состояния**.

Если **начальная конфигурация s мало отличается** от одного из таких аттракторов s^* , то сеть быстро эволюционирует к этому аттрактору:

$$s \rightarrow s^*$$

При этом **изменяется состояние** только небольшого числа нейронов.

Интерпретация такого перехода: исходное состояние s содержит частичную, неполную информацию, которая, однако, достаточна для восстановления полной информации, кодируемой состоянием s^* .

Пример:

В*нец*я



Венеция

Важное свойство такой памяти, представленной **набором аттракторов** сети — **распределенность**.

Все нейроны сети участвуют в кодировании **всех состояний памяти**.

Небольшие искажения отдельных весов сети не сказываются на содержании памяти, что повышает **устойчивость памяти к помехам**.

Модели с обратными связями (XXXIV)

Сеть Хопфилда как ассоциативная память – 11

Обучение сети (1)

Пример задачи для ассоциативной памяти,
реализуемой с помощью сети Хопфилда

Построить сеть, **аттракторы** которой соответствовали бы векторам, кодирующих бинарные **изображения подписей** различных людей.

Практически невозможно совершенно одинаково расписаться дважды, модель Хопфилда позволяет решить задачу распознавания подписей, несмотря на их естественную **вариативность**.

Если число различных типов подписей, которые должна распознавать сеть, равно **P** и **образцы** в некотором смысле типичных, наиболее вероятных или усредненных подписей различных людей **кодируются векторами** $\sigma^{(n)}$, $n = 1, \dots, P$, то желательно, чтобы именно эти векторы кодировали и **аттракторы сети**, предназначенной для классификации.

Модели с обратными связями (XXXV)

Сеть Хопфилда как ассоциативная память – 12

Обучение сети (2)

Хопфилд предложил использовать для решения задачи формирования ассоциативной памяти **хеббовское правило** построения межнейронных связей:

$$w_{ij} = \frac{1}{N} \sum_n \sigma_i^{(n)} \sigma_j^{(n)}, \quad i \neq j, \quad w_{ii} = 0, \quad i, j = 1, \dots, N$$

Это правило **гарантирует стационарность** произвольно выбранных векторов $\sigma^{(n)}$ в случае, когда их число P не превосходит примерно 5% от общего числа нейронов N .

При больших значениях P некоторые из запоминаемых векторов $\sigma^{(n)}$ теряют **свойство стационарности**, а при превышении некоторого **критического значения**, называемого **емкостью памяти** ($P \approx 0.14N$) стационарные состояния сети теряют всякую связь с ними, и сеть переходит из режима запоминания в **режим спинового стекла**, для которого характерно наличие очень большого числа аттракторов, **далеких от любых запоминаемых векторов**.

Аттракторы, **не совпадающие** с векторами $\sigma^{(n)}$ — **ложная память** (паразитная память, химеры и т.п.).

Модели с обратными связями (XXXVI)

Сеть Хопфилда как ассоциативная память – 13

Разобучение сети

Процедура уменьшения доступа к состояниям ложной памяти — **разобучение**.

Разобучение применяется к уже обученной сети, в пространстве которой есть **ложные состояния**.

Сети **многократно предъявляются** в качестве начальных состояний **случайно сгенерированные векторы**, затем прослеживается их эволюция вплоть до стационарного состояния σ^* , которое может принадлежать **как истинной, так и ложной памяти**.

После этого **связи в сети** модифицируются следующим образом:

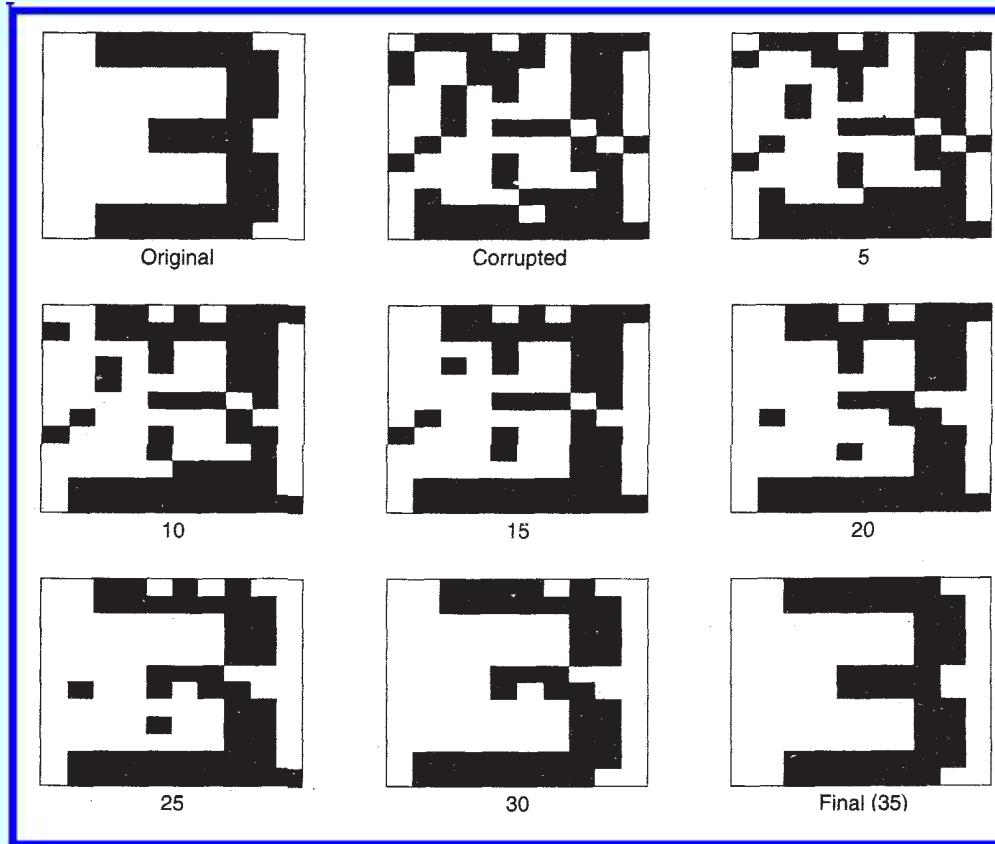
$$\delta w_{ij} = -\epsilon \sigma_i^* \sigma_j^*, \quad i \neq j$$

Здесь $\epsilon > 0$ — небольшая константа.

Для сети, обученной по правилу Хебба на наборе случайных векторов, увеличивается и выравнивается доступность состояний, соответствующих запоминаемым образам, а также снижается доступность состояний ложной памяти.

Модели с обратными связями (XXXVII)

Примеры использования сети Хопфилда – 1

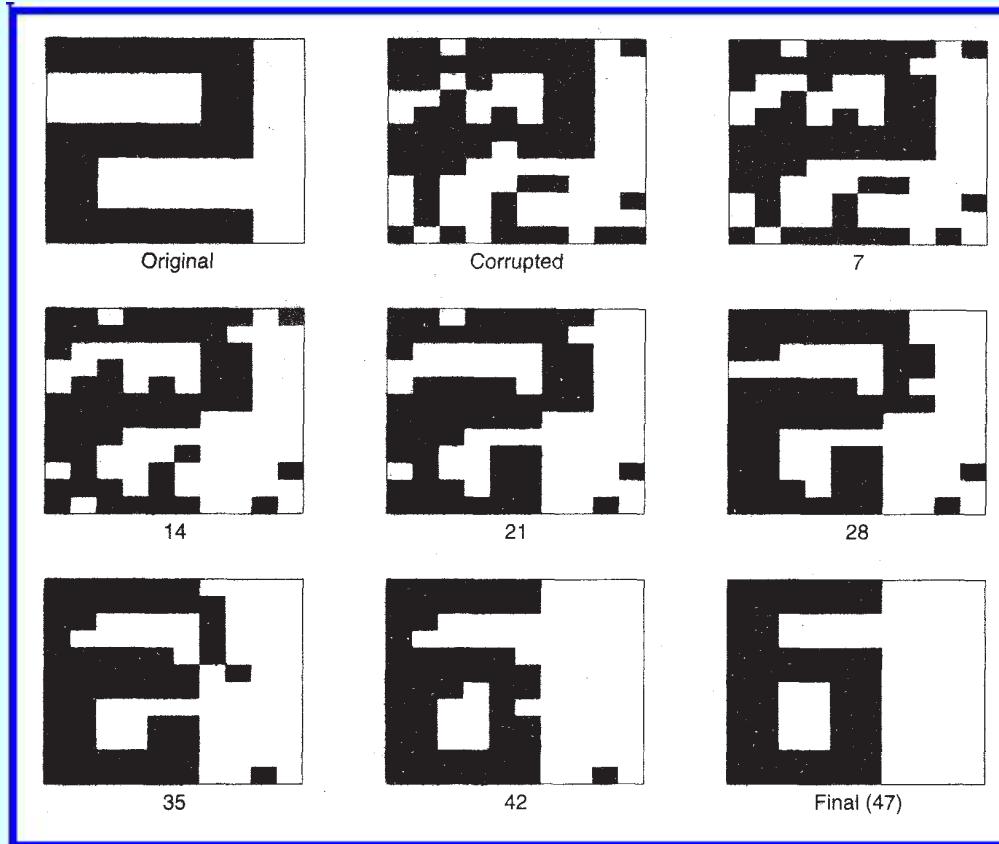


Корректное восстановление искаженного входного паттерна (цифра “3”)

Источник: *Хайкин С.* Нейронные сети: Полный курс. – 2-изд. – М.: Вильямс, 2006. – 1104 с.
(Гл.14, рис.14.18, с.878).

Модели с обратными связями (XXXVIII)

Примеры использования сети Хопфилда – 2

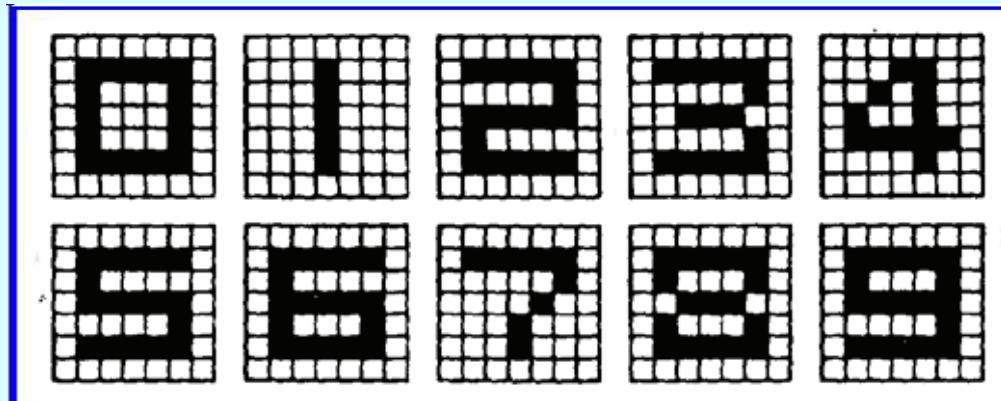


Некорректное восстановление искаженного входного паттерна (цифра “2”)

Источник: [Хайкин С.](#) Нейронные сети: Полный курс. – 2-изд. – М.: Вильямс, 2006. – 1104 с.
(Гл.14, рис.14.19, с.879).

Модели с обратными связями (XXXIX)

Примеры использования сети Хопфилда – 3

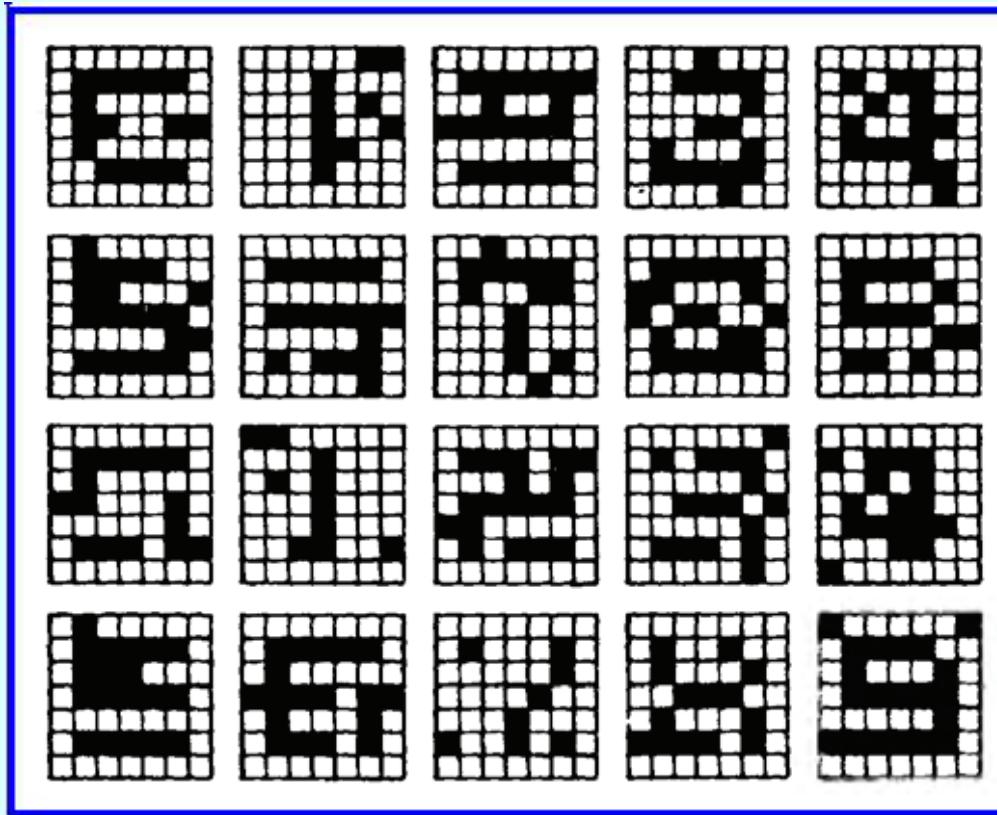


Паттерны-эталоны, использованные для обучения сети Хопфилда

Источник: *Осовский С.* Нейронные сети для обработки информации: Пер. с польск. – М.: Финансы и статистика, 2002. – 344 с. (Глава 7, рис.7.2, с.183).

Модели с обратными связями (XL)

Примеры использования сети Хопфилда – 4



Зашумленные паттерны, использованные для тестирования сети Хопфилда

Источник: Осовский С. Нейронные сети для обработки информации: Пер. спольск. – М.: Финансы и статистика, 2002. – 344 с. (Глава 7, рис.7.3, с.183).

Модели с обратными связями (ХЛІ)

Развитие модели Хопфилда – 1

Недостатки модели Хопфилда (1)

Типичные проблемы, связанные с сетями Хопфилда:

- 1) **проблема информационной емкости** (сколько паттернов-эталонов можно записать в такую сеть, а затем воспроизвести их?);
- 2) **проблема качества воспроизведения** (какова будет доля ошибок в выходных паттернах в сравнении с воспроизводимыми эталонами?);
- 3) **проблема размеров областей притяжения** (насколько сильно может быть искажен входной паттерн в сравнении с эталоном, чтобы сохранить свойство воспроизводимости?).

Модели с обратными связями (XLII)

Развитие модели Хопфилда – 2

Недостатки модели Хопфилда (2)

Сеть Хопфилда размерности N имеет 2^N состояний, однако ее **максимальная информационная емкость** оказывается значительно меньшей.

Невысокая информационная емкость:

Емкость памяти стандартной модели Хопфилда составляет около **14%** от размерности задачи N .

«Катастрофа памяти»:

Память сети Хопфилда **разрушается при ее переполнении**: если записать в межсвязи число паттернов, превосходящее **$0.14N$** , память сети полностью разрушится.

Модели с обратными связями (XLIII)

Развитие модели Хопфилда – 3

**Пути преодоления недостатков модели Хопфилда:
Невысокая информационная емкость (1)**

Последние 15 лет отмечены интересом к **моделям ассоциативной памяти** с *q*-нарными нейронами.

Все в этих моделях подобно модели Хопфилда, но число состояний *q*, в которых могут находиться нейроны, **больше 2**.

Один из наиболее интересных вариантов *q*-нарных моделей — **параметрические нейронные сети**.

Модели с обратными связями (XLIV)

Развитие модели Хопфилда – 4

Пути преодоления недостатков модели Хопфилда:
Невысокая информационная емкость (2)

Параметрическая нейронная сеть опирается на **параметрический нейрон**,
состояние которого описывается следующим образом:

$$\vec{x}_i = x_i \vec{e}_{l_i}, \text{ где } x_i = \pm 1, \vec{e}_l = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^q, \quad \left\{ \begin{array}{l} i = 1, \dots, N; \\ l = 1, \dots, q; \\ 1 \leq l_i \leq q. \end{array} \right.$$

Так как $x_i = \pm 1$, нейроны в ПНС могут находиться в **$2q$ различных состояниях**.

Состояние сети как целого задается набором N таких q -мерных векторов $\mathbf{X} = (\vec{x}_1, \dots, \vec{x}_N)$.

При $q = 1$ ПНС переходит в **стандартную модель Хопфилда**: векторы \vec{x}_i превращаются в обычные бинарные переменные $x_i = \pm 1$.

Модели с обратными связями (XLV)

Развитие модели Хопфилда – 5

Пути преодоления недостатков модели Хопфилда:
Невысокая информационная емкость (3)

Асимптотическая оценка емкости памяти ПНС при $N \gg 1$

Начальное состояние сети — искаженный m -й паттерн:

$$\widetilde{X}^m = (a_1 \widehat{b}_1 \vec{x}_1^m, a_2 \widehat{b}_2 \vec{x}_2^m, \dots, a_N \widehat{b}_N \vec{x}_N^m).$$

Независимые случайные величины $\{a_i\}_1^N$ и $\{\widehat{b}_i\}_1^N$ задают **мультипликативный шум**.

Случайная величина a_i с вероятностью a принимает значение -1 , а с вероятностью $(1 - a)$ — значение 1 (в оптической терминологии — *амплитудный шум*).

Случайный оператор \widehat{b}_i с вероятностью b изменяет состояние i -го нейрона на другое, а с вероятностью $(1 - b)$ оставляет вектор \vec{x}_i^m неизменным (в оптической терминологии — *частотный шум*).

Модели с обратными связями (XLVI)

Развитие модели Хопфилда – 6

Пути преодоления недостатков модели Хопфилда:
Невысокая информационная емкость (4)

Асимптотическая оценка емкости памяти ПНС при $N \gg 1$

Вероятность ошибки распознавания паттерна X^m :

$$\Pr_{err} = N \exp \left[-\frac{N(1-2a)^2}{2p} \cdot q^2(1-b)^2 \right].$$

Асимптотически достижимая емкость памяти ПНС:

$$p_c = \frac{N(1-2a)^2}{2 \ln N} \cdot q^2(1-b)^2.$$

При $q = 1$ эти выражения превращаются в известные результаты для модели Хопфилда (в этом случае $b = 0$).

С ростом q экспоненциально уменьшается вероятность ошибки распознавания — существенно **растет помехоустойчивость сети**.

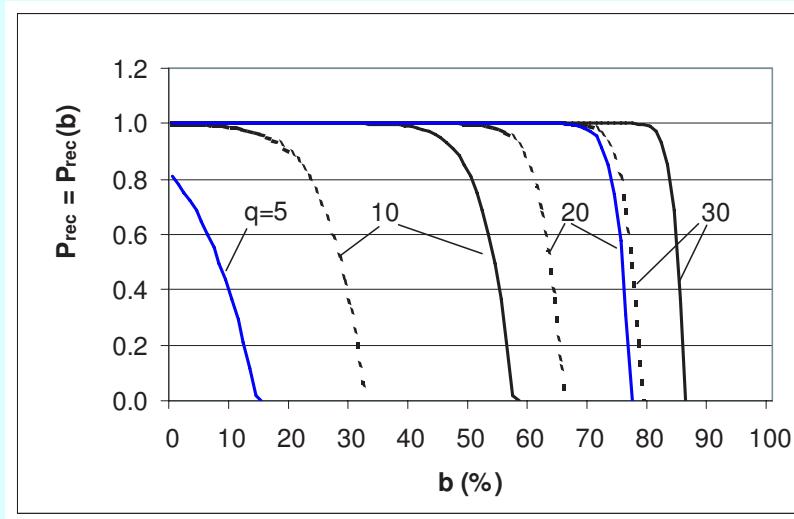
Одновременно, пропорционально q^2 растет и **емкость памяти**.

В отличие от модели Хопфилда, оказывается возможным эффективное запоминание большего, чем N , числа паттернов p .

Модели с обратными связями (XLVII)

Развитие модели Хопфилда – 7

Пути преодоления недостатков модели Хопфилда: Невысокая информационная емкость (5)



Зависимость вероятности правильного распознавания $P_{rec} = 1 - Pr_{err}$ для различных значений q от величины частотного шума $b = b \cdot 100\%$, $b \in [0, 1]$ при числе паттернов, вдвое большем числа нейронов ($p = 2N$) и $a = 0$ (сплошная линия)

Для $q = 20$ практически со стопроцентной вероятностью будет правильно восстановлен любой паттерн, зашумленный не более, чем на 70%, а для $q = 30$ – любой паттерн, зашумленный не более, чем на 85%.

Если уровень шума b меньше критического значения $b_c = 1 - 2/q\sqrt{p/N}$, ПНС практически со стопроцентной вероятностью восстановит зашумленный паттерн.

Если же $b > b_c$, вероятность правильного распознавания паттерна стремится к нулю.

Источник: Крыжановский Б.В., Литинский Л.Б. Векторные модели ассоциативной памяти // V Всероссийская научно-техническая конференция «Нейроинформатика-2003»: Лекции по нейроинформатике. Часть 1. – М.: МИФИ, 2003. – с. 82.

Модели с обратными связями (XLVIII)

Развитие модели Хопфилда – 8

Пути преодоления недостатков модели Хопфилда:
Невысокая информационная емкость (6)

Разреженное кодирование

Один из возможных подходов, позволяющих увеличить информационную емкость сети Хопфилда — **разреженное кодирование**.

При таком кодировании $n \ll N$, т. е. количество **активных нейронов** n в записанных паттернах-эталонах **много меньше** общего количества N нейронов в сети.

В предельном случае, когда $n/N \rightarrow 0$, оценка максимального числа запоминаемых паттернов составляет $0.72N$.

Модели с обратными связями (XLIX)

Развитие модели Хопфилда – 9

Пути преодоления недостатков модели Хопфилда:
«Катастрофа памяти» (1)

Загрузка сети Хопфилда — отношение $\alpha = M/N$ числа паттернов M к числу N нейронов.

Критическое число паттернов M_c для данной размерности N :

$$M_c = \alpha_c N, \quad \text{где } \alpha_c = 0.138$$

Пока $M < M_c$, в небольшой окрестности каждого паттерна обязательно имеется локальный минимум функции ошибки и его перекрытие с паттерном **близко к 1**.

То есть, пока $M < M_c$, модель Хопфилда **работает как память**.

Когда число паттернов M становится больше критического M_c , они **перестают распознаваться** сетью.

Модели с обратными связями (L)

Развитие модели Хопфилда – 10

Пути преодоления недостатков модели Хопфилда:
«Катастрофа памяти» (2)

Если $M > M_c$, паттерны перестают распознаваться сетью.

Локальные минимумы **скачком удаляются** от паттернов, их **перекрытие** с паттернами становится **близко к 0**.

Происходит **катастрофическое разрушение памяти** — сеть перестает работать как запоминающее устройство.

Такое поведение сети Хопфилда является ее **дефектом**.

Память системы **не должна разрушаться** при ее переполнении.

Обновление памяти должно происходить **путем частичной замены, а не путем полного забывания** всего материала, накопленного ранее.

Иными словами, попытка записать в систему паттерны **сверх установленного лимита** **не должна приводить к разрушению памяти**.

Модели с обратными связями (LI)

Развитие модели Хопфилда – 11

**Пути преодоления недостатков модели Хопфилда:
«Катастрофа памяти» (3)**

В стандартной модели Хопфилда **этап обучения и рабочая стадия четко разделены.**

На этапе обучения с помощью исходных паттернов рассчитывается матрица связей.

Прежде чем дописывать в матрицу связей дополнительные паттерны, **необходимо убедиться** в том, что новое число паттернов не превышает **критического значения $M_c \approx 0.138N$** , в противном случае память сети будет разрушена.

Память должна работать, даже если новая информация записывается в нее **непрерывно, процесс обучения никогда не прерывается.**

Данный дефект можно устранить, снабдив паттерны **различными весовыми множителями.**

Эти веса интерпретируются как **частоты появления паттернов** на стадии обучения сети.

Модели с обратными связями (LII)

Развитие модели Хопфилда – 12

**Пути преодоления недостатков модели Хопфилда:
«Катастрофа памяти» (4)**

Веса связей сети **модифицируются** по мере поступления новых паттернов.

Распознаются только те паттерны, для которых веса λ больше некоторого критического значения λ_c .

Пусть вес некоторого паттерна меньше λ_c , но нам надо, чтобы данный паттерн сетью распознавался.

Достаточно **увеличить его вес**, сделать его больше λ_c и паттерн будет сетью распознаваться.

При этом **могут перестать распознаваться** некоторые другие паттерны, веса которых лишь слегка превосходили λ_c .

Такой **способ обновления** памяти **за счет вытеснения** одних паттернов **другими** не противоречит здравому смыслу и отвечает общей концепции устройства человеческой памяти.

Модели с обратными связями (LIII)

Развитие модели Хопфилда – 13

Сеть Хопфилда с q -нарными нейронами

Крыжановский Б. В., Литинский Л. Б. Векторные модели ассоциативной памяти // V Всероссийская научно-техническая конференция «Нейроинформатика-2003»: Лекции по нейроинформатике. Часть 1. – М.: МИФИ, 2003. – с. 72–85.

Литинский Л. Б. Параметрические нейронные сети и другие архитектуры на их основе (обзор работ) // VIII Всероссийская научно-техническая конференция «Нейроинформатика-2006»: Лекции по нейроинформатике. – М.: МИФИ, 2006. – с. 231–243.

Крыжановский Б. В. О топологии потенциальной поверхности и вероятностях обнаружения локальных минимумов в задачах бинарной оптимизации // X Всероссийская научно-техническая конференция «Нейроинформатика-2008»: Лекции по нейроинформатике. Часть 2. – М.: МИФИ, 2008. – с. 12–36.

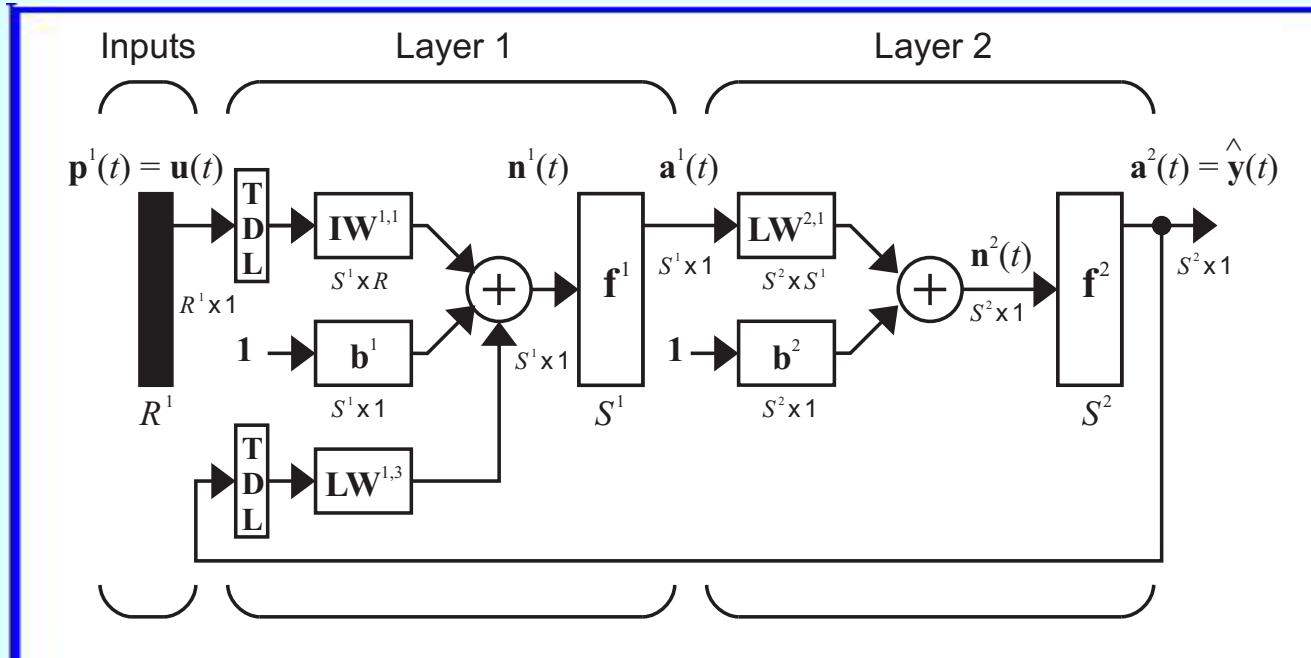
Карандашев Я. М., Крыжановский Б. В., Литинский Л. Б. Взвешенные паттерны и устранение «катастрофы памяти» в модели Хопфилда // XIV Всероссийская научно-техническая конференция «Нейроинформатика-2012»: Лекции по нейроинформатике. – М.: МИФИ, 2012. – с. 152–189.

Сеть Хопфилда с разреженным кодированием

Фролов А. А., Гусек Д., Муравьев И. П. Информационная эффективность ассоциативной памяти типа Хопфилда с разреженным кодированием // V Всероссийская научно-техническая конференция «Нейроинформатика-2003»: Лекции по нейроинформатике. Часть 1. – М.: МИФИ, 2003. – с. 28–71.

Модели с обратными связями (LIV)

Модель NARX – 1



NARX – Nonlinear AutoRegressive network with eXogeneous inputs

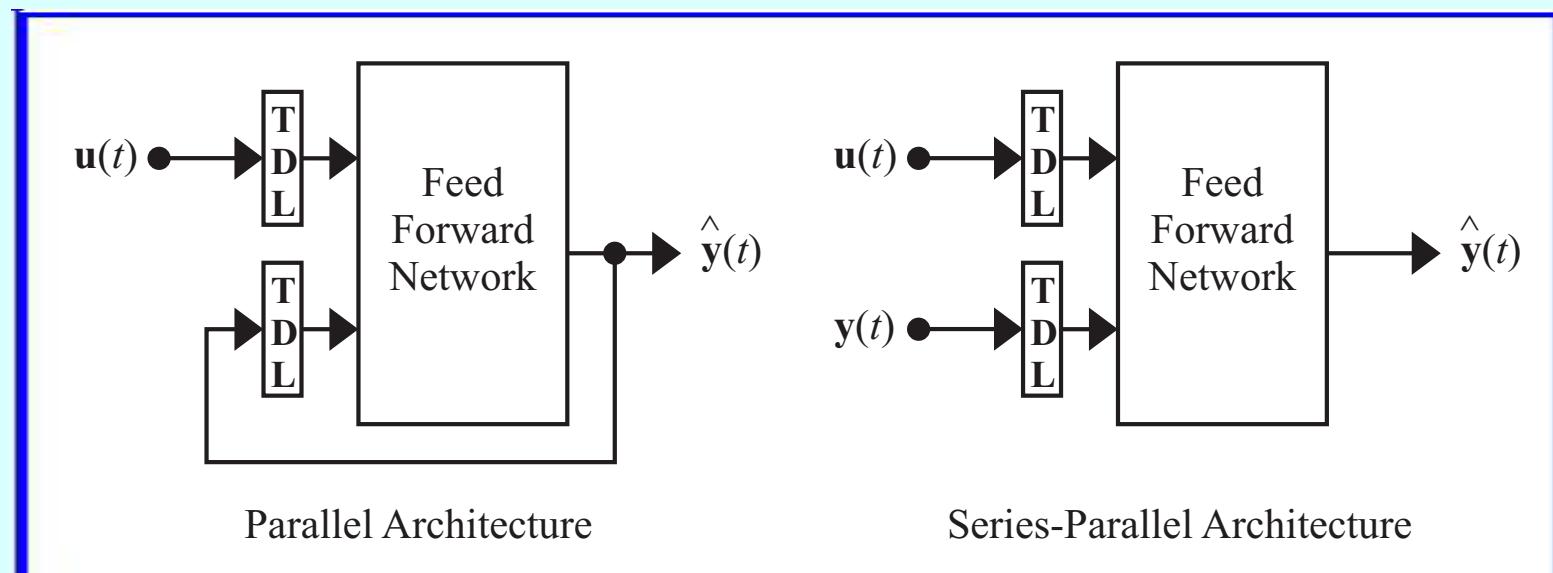
(нелинейная авторегрессионная сеть с внешними входами)

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u))$$

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 6-18).

Модели с обратными связями (LV)

Модель NARX – 2



Параллельная и последовательно-параллельная схема обучения сети NARX

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 6-19).

Модели с обратными связями (LVI)

NARX в задачах моделирования и управления – 1

**Пример: Управление движением
современных и перспективных самолетов**

Управление движением современных и перспективных самолетов приходится обеспечивать в условиях значительных и разнообразных **неопределенностей**.

Источники этих неопределенностей:

- значения параметров и характеристик самолета,
- реализуемые режимы полета,
- воздействия внешней среды.

Кроме того, в ходе полета могут возникать разнообразные **нештатные ситуации, изменяющие динамику** самолета, в том числе:

- отказы** систем и оборудования,
- повреждения** конструкции.

Эти неопределенностей, включая вызванные нештатными ситуациями, **необходимо парировать** за счет **реконфигурации** системы управления и органов управления самолета.

Модели с обратными связями (LVII)

NARX в задачах моделирования и управления – 2

Адаптация как средство управления в условиях неопределенности

Ситуация, в которой оказывается самолет в каждый текущий момент времени, из-за неопределенностей может значительно меняться **непредсказуемым заранее** образом.

Система управления самолета должна быть в состоянии **эффективно приспосабливаться** к этим изменениям за счет оперативного изменения параметров и/или структуры используемых законов управления.

Удовлетворить этому требованию позволяет аппарат теории **адаптивного управления**.

К числу наиболее популярных относятся следующие две **схемы адаптивного управления**:

- адаптивное управление с **эталонной** моделью,
- адаптивное управление с **прогнозирующей** моделью.

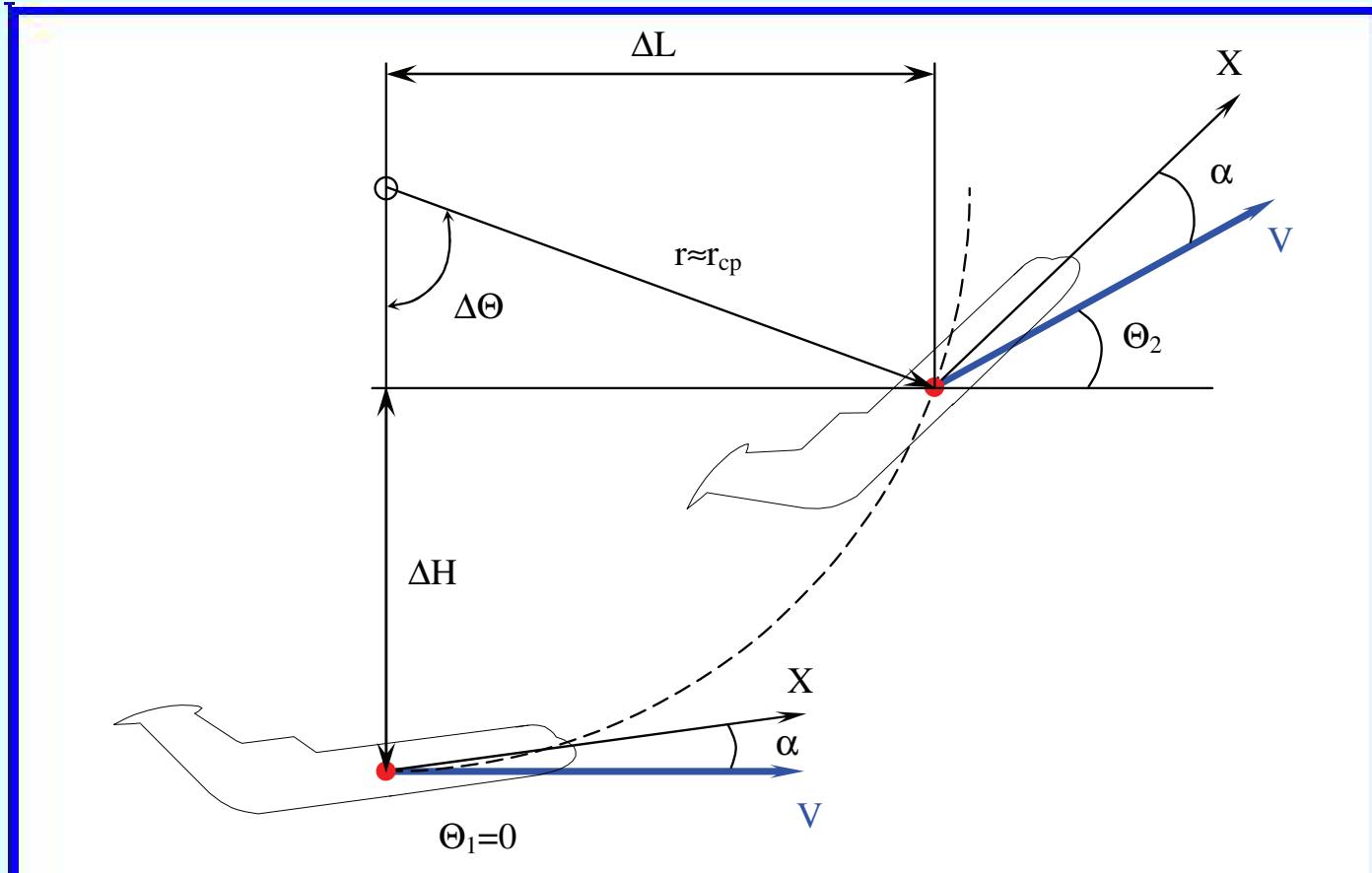
Один из эффективных подходов к реализации концепций адаптивности, особенно для нелинейного объекта, основан на методах и средствах **нейросетевого моделирования и управления**.

Инструмент решения указанных задач — **многослойные сети персептронного типа** с обратными связями и с линиями задержки на входах, обучаемые с использованием динамических схем.

Модели с обратными связями (LVIII)

NARX в задачах моделирования и управления – 3

Продольное движение самолета



V — скорость полета, α — угол атаки, Θ — угол наклона траектории,
 $\vartheta = \Theta + \alpha$ — угол тангажа, $\dot{\vartheta} = \omega_z$ — угловая скорость тангажа

Модели с обратными связями (LIX)

NARX в задачах моделирования и управления – 4

Традиционная модель объекта управления

Рассмотрим решение задачи нелинейного адаптивного отказоустойчивого управления на примере **продольного углового движения** маневренного самолета.

Будем описывать это движение с помощью **математической модели**, традиционной для задач динамики полета самолетов:

$$\begin{aligned}\dot{\alpha} &= \omega_z - \frac{qS}{mV} C_{ya}(\alpha, \omega_z, \varphi) + \frac{g}{V}, \\ \dot{\omega}_z &= \frac{qSb_A}{J_{zz}} m_z(\alpha, \omega_z, \varphi), \\ T^2 \ddot{\varphi} &= -2T\zeta\dot{\varphi} - \varphi + \varphi_{act},\end{aligned}$$

где α — угол атаки, град; ω_z — угловая скорость тангажа, град/с; φ — угол отклонения управляемого стабилизатора, град; C_{ya} — коэффициент подъемной силы; m_z — коэффициент момента тангажа; m — масса самолета, кг; V — воздушная скорость, м/с; $q = \rho V^2 / 2$ — скоростной напор; ρ — плотность воздуха, кг/м³; g — ускорение силы тяжести, м/с²; S — площадь крыла, м²; b_A — средняя аэродинамическая хорда крыла, м; J_{zz} — момент инерции самолета относительно боковой оси, кг·м²; безразмерные коэффициенты C_{ya} и m_z являются нелинейными функциями своих аргументов; T , ζ — постоянная времени и коэффициент относительного демпфирования привода, φ_{act} — командный сигнал на привод (ограничивается $\pm 25^\circ$).

В данной модели величины α , ω_z , φ и $\dot{\varphi}$ — это **состояния** объекта управления, величина φ_{act} — **управление**.

Модели с обратными связями (LX)

NARX в задачах моделирования и управления – 5

Нейросетевая идентификация объекта управления: Общая схема

В качестве подзадачи синтеза адаптивного закона управления необходимо решать задачу **идентификации** для рассматриваемого объекта управления, т.е. требуется получить **нейросетевую аппроксимацию** исходной математической модели движения самолета.

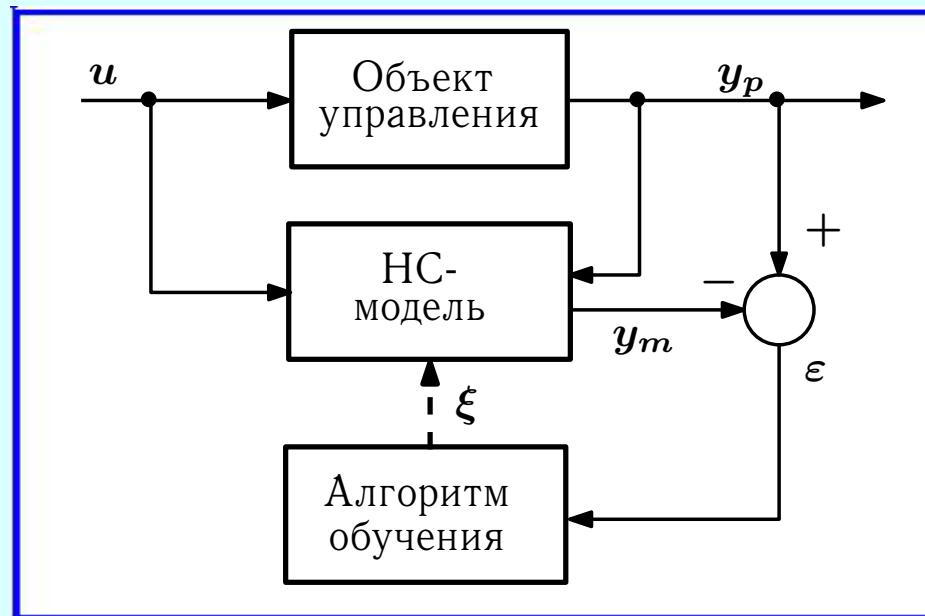
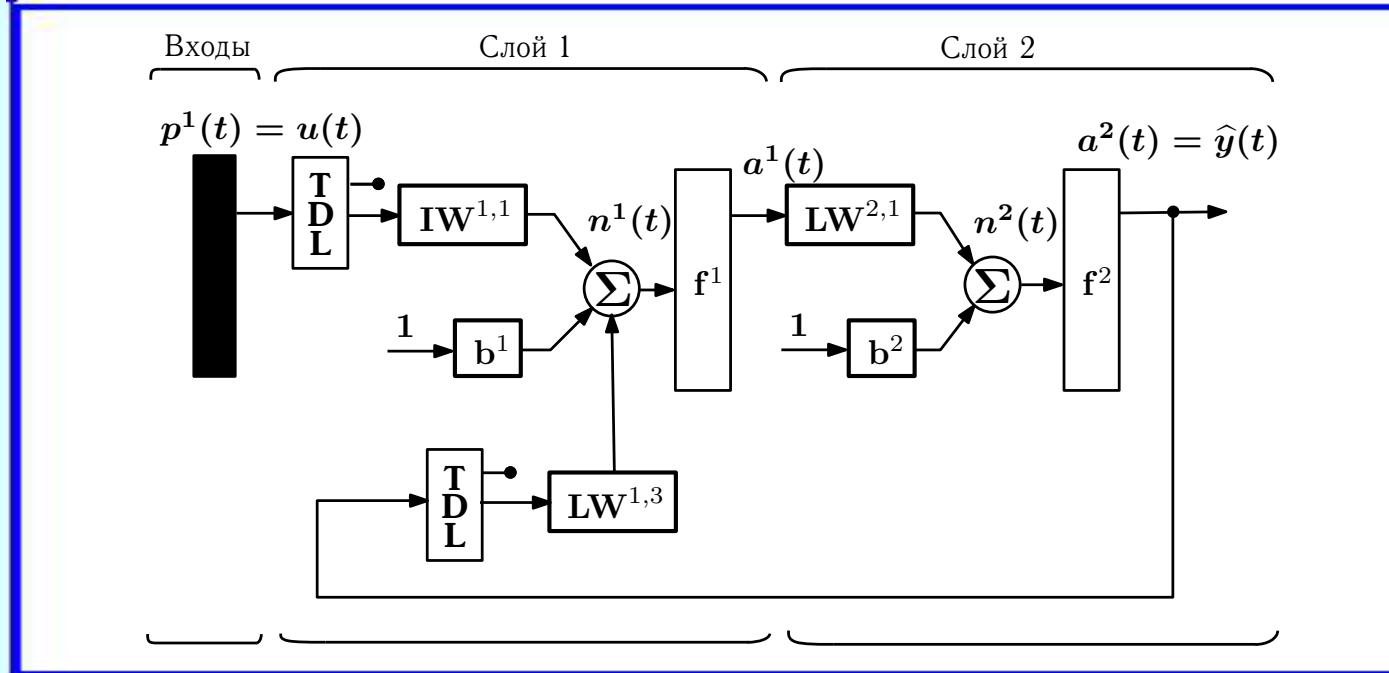


Схема нейросетевой идентификации объекта управления: u — управление, y_p — выход объекта управления, y_m — выход нейросетевой модели объекта управления; ε — расхождение между выходами объекта управления и НС-модели; ξ — корректирующее воздействие.

Модели с обратными связями (LXI)

NARX в задачах моделирования и управления – 6

Нейросетевая идентификация объекта управления: NARX-модель



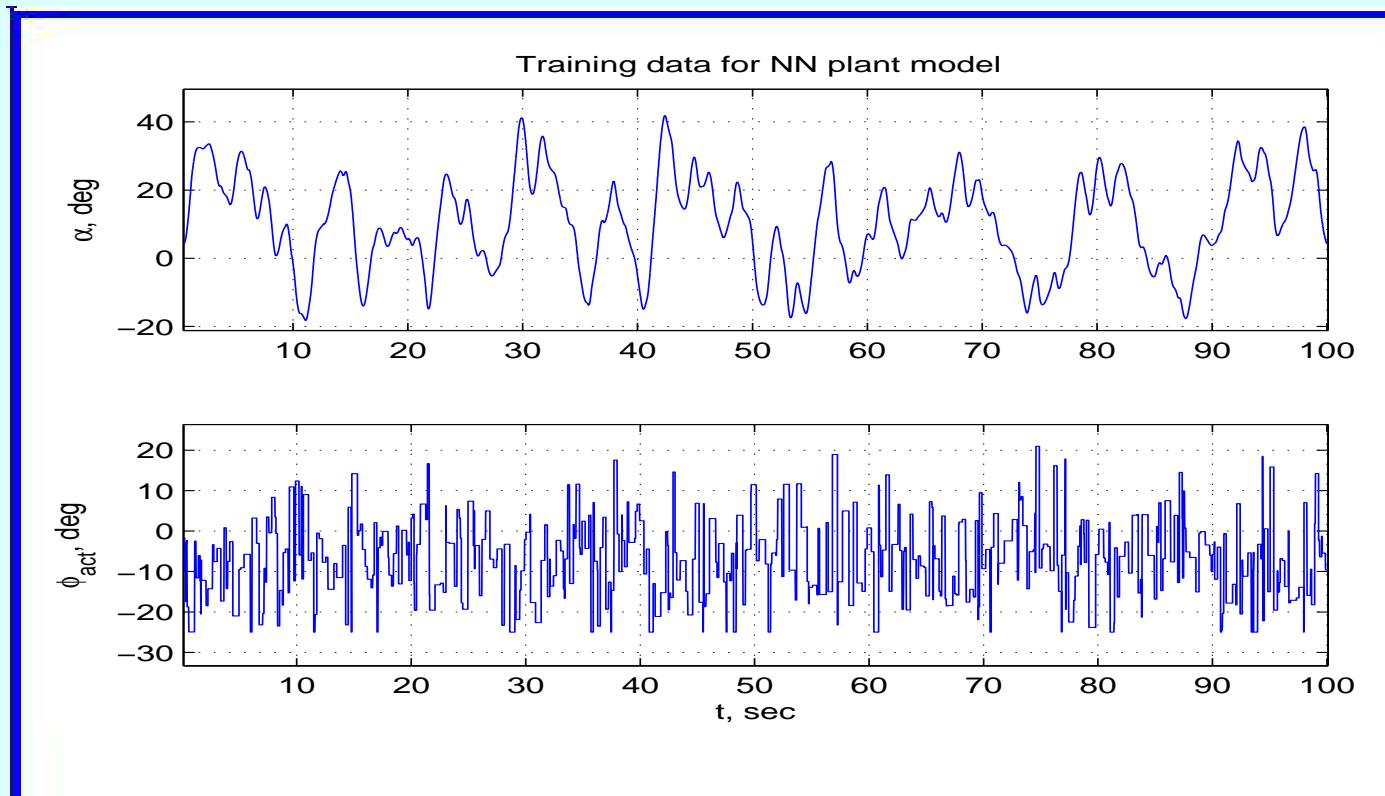
Структурная схема нейросетевой NARX-модели объекта управления

TDL — линия задержки; **IW** — матрица синаптических весов связей между входным и первым обрабатывающим слоем НС; **LW** — матрица синаптических весов связей между обрабатывающими слоями НС; **b** — набор смещений слоя НС; **f** — набор активационных функций слоя НС; **Σ** — набор сумматоров слоя НС; **$n(t)$** — набор скалярных выходов сумматоров; **$a(t)$** — набор скалярных выходов активационных функций; **$p^1(t) = u(t)$** — входной сигнал; **$\hat{y}(t)$** — выход НС-модели.

Модели с обратными связями (LXII)

NARX в задачах моделирования и управления – 7

Нейросетевая идентификация объекта управления:
Формирование НС-модели

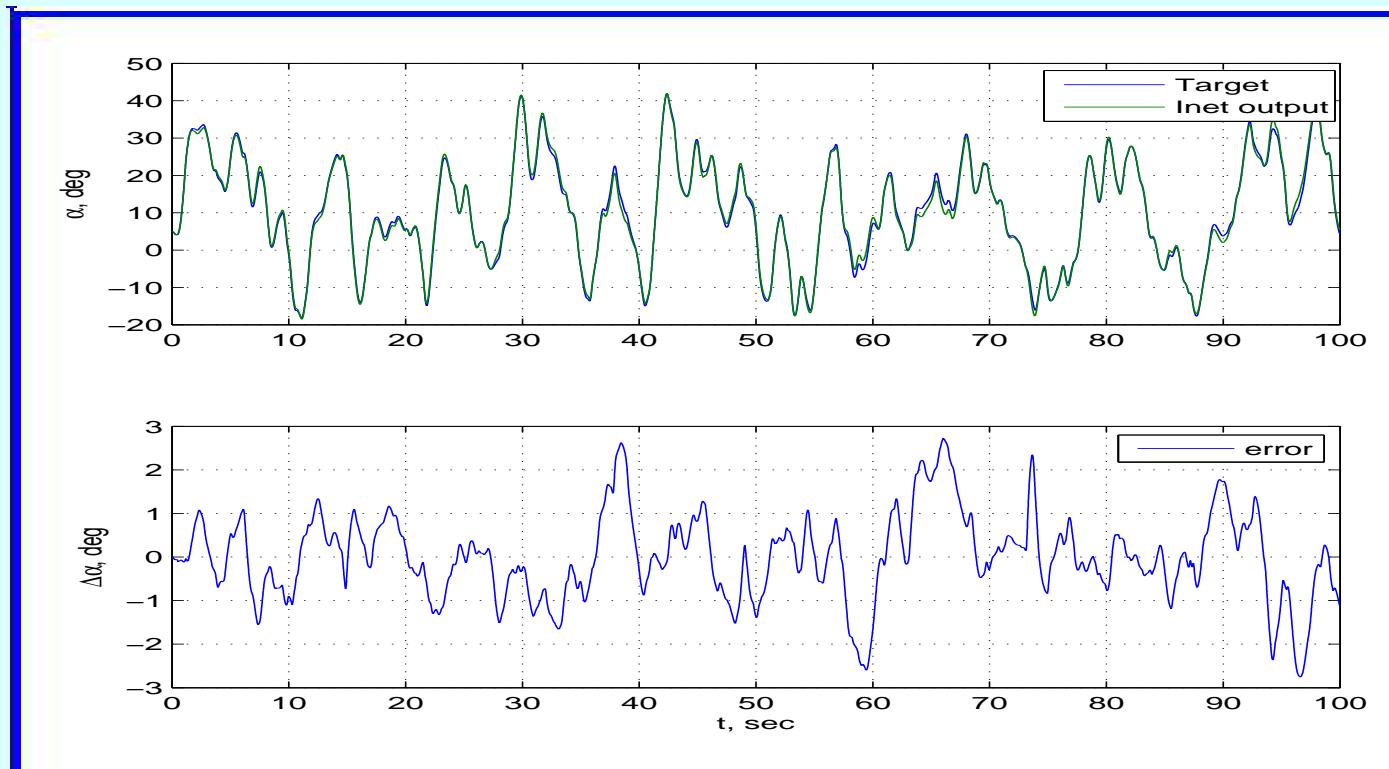


Обучающая выборка для нейросетевой модели объекта управления, режим полета с индикаторной скоростью $V_i = 500$ км/ч

Модели с обратными связями (LXIII)

NARX в задачах моделирования и управления – 8

Нейросетевая идентификация объекта управления:
Тестирование НС-модели

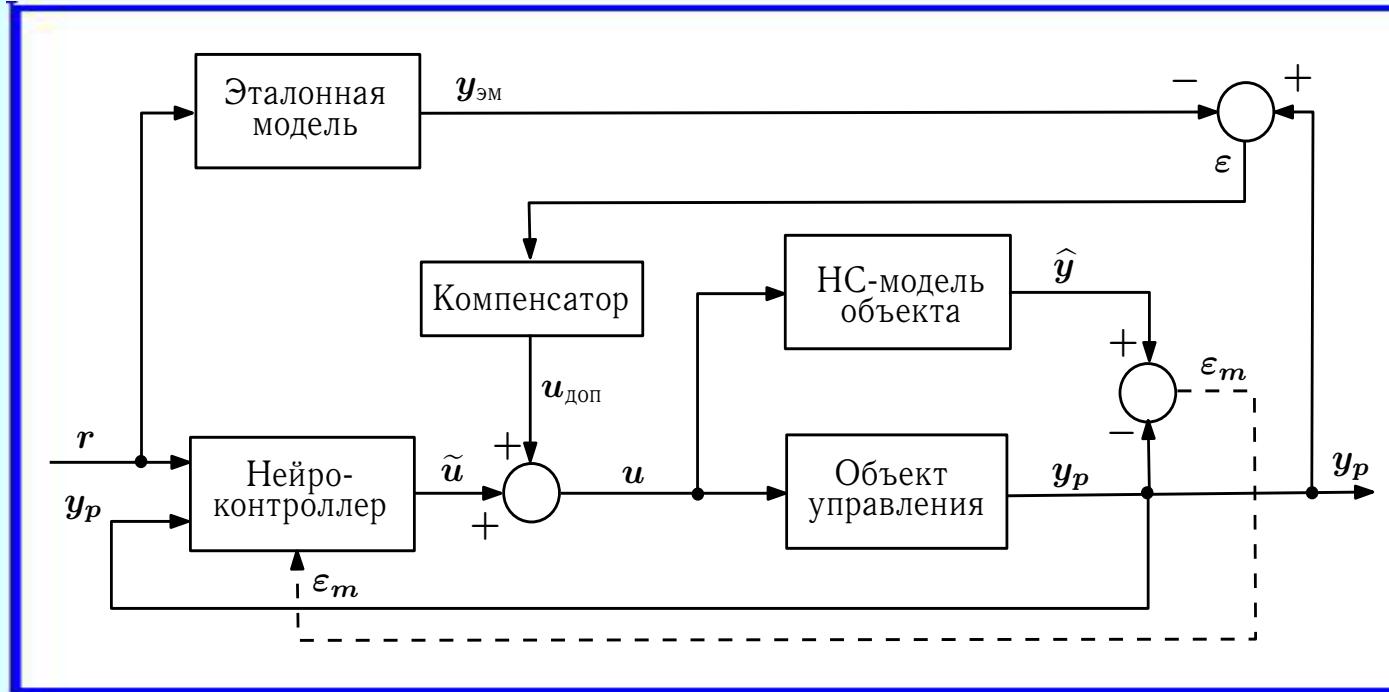


Проверка работоспособности замкнутой нейросетевой модели объекта управления, режим полета с индикаторной скоростью $V_i = 500$ км/ч

Модели с обратными связями (LXIV)

NARX в задачах моделирования и управления – 9

Адаптивное управление с эталонной моделью (1)



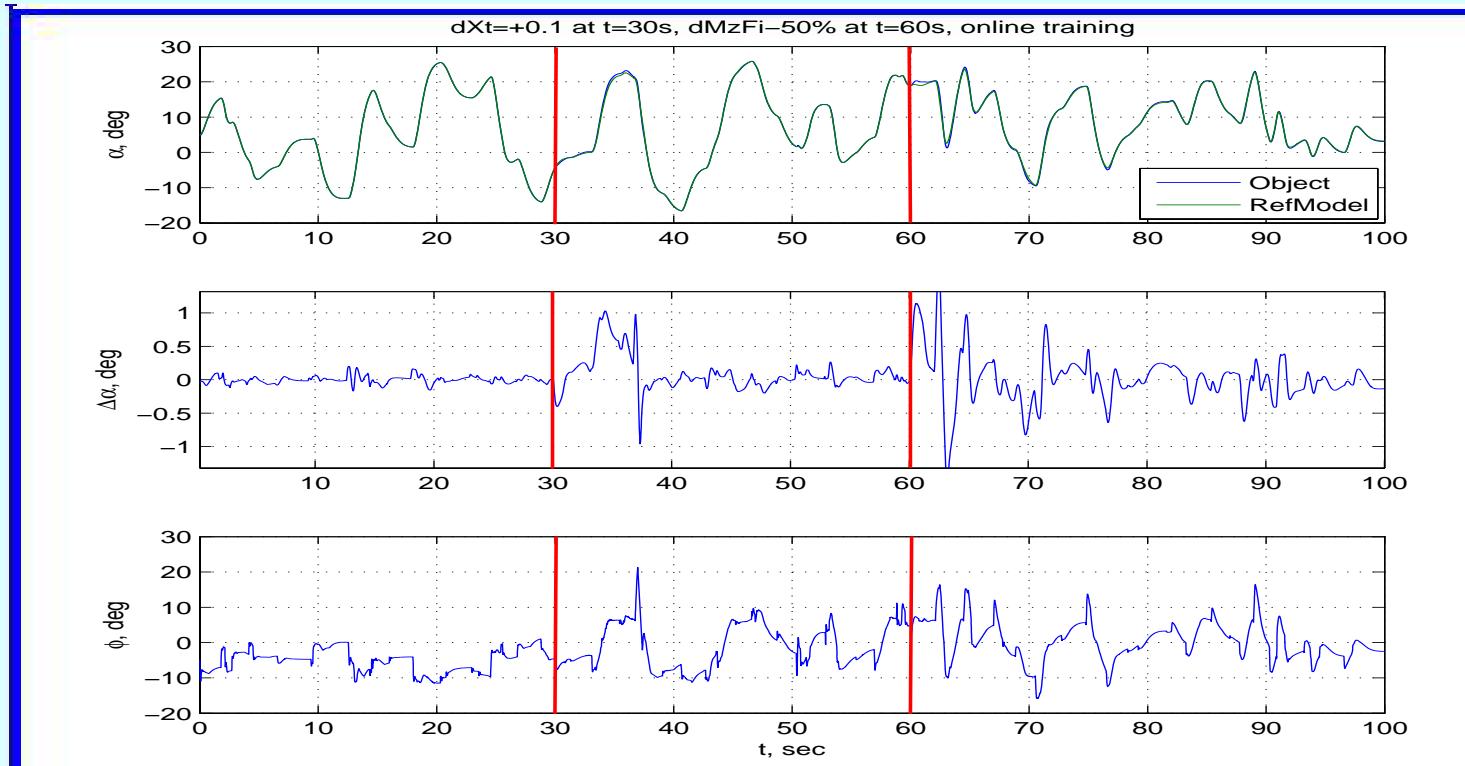
Общая схема нейросетевого адаптивного управления с эталонной моделью

Здесь: \tilde{u} — управление на выходе нейроконтроллера, $u_{\text{доп}}$ — добавочное управление от компенсатора, u — результирующее управление, y_p — выход объекта управления, \hat{y} — выход нейросетевой модели объекта управления; $y_{\text{эм}}$ — выход эталонной модели; ε — расхождение между выходами объекта управления и эталонной модели; ε_m — расхождение между выходами объекта управления и НС-модели; r — задающее воздействие.

Модели с обратными связями (LXV)

NARX в задачах моделирования и управления – 10

Адаптивное управление с эталонной моделью (2)

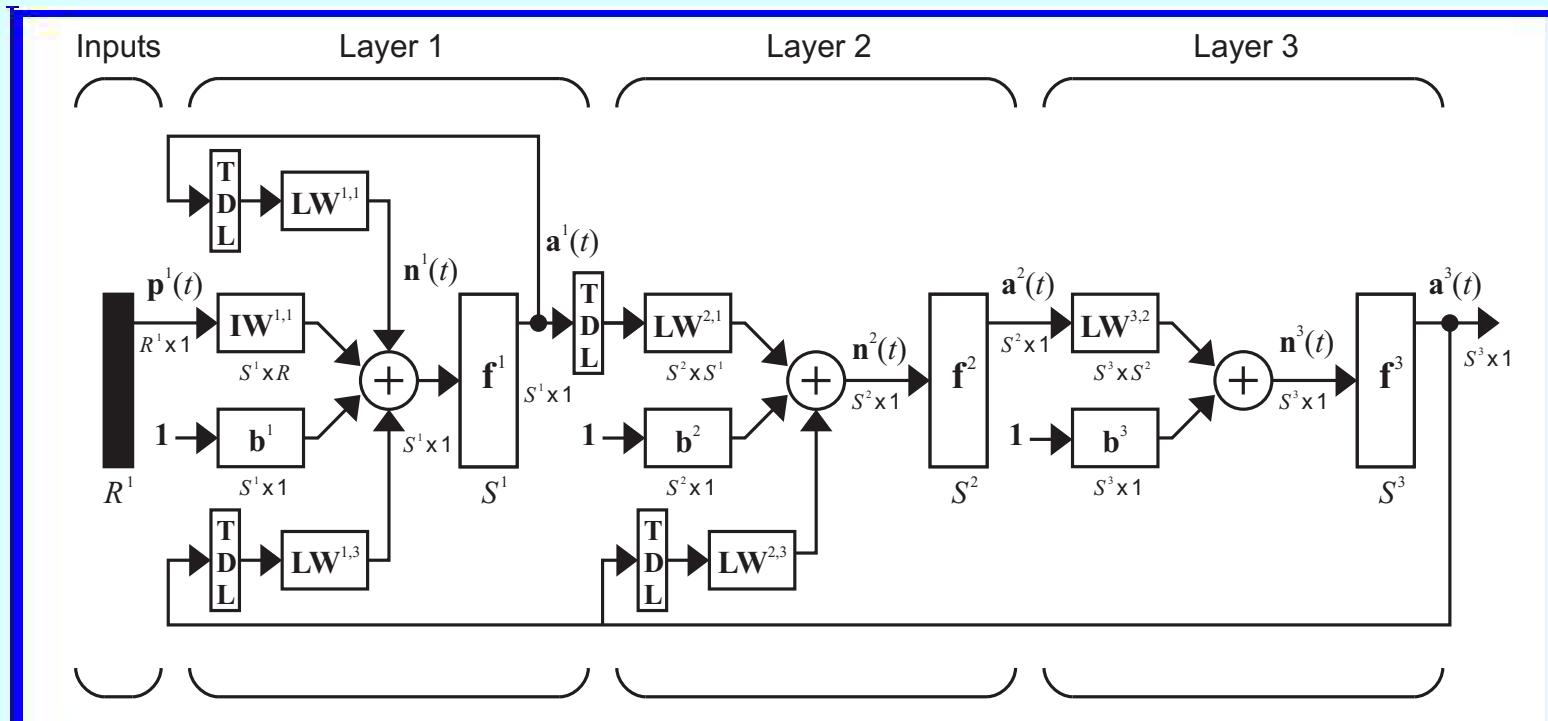


Результаты вычислительного эксперимента для системы управления с эталонной моделью и компенсатором (самолет F-16, режим полета с индикаторной скоростью $V_{ind} = 700 \text{ км/ч}$). Адаптация к изменению динамики объекта управления: **смещение центровки** на 10% назад ($t = 30 \text{ с}$), 50% **уменьшение эффективности** органа управления ($t = 60 \text{ с}$). Обозначения: α — угол атаки, град.; $\Delta\alpha$ — ошибка отслеживания заданного угла атаки, град.; ϕ — угол отклонения стабилизатора, град.; t — время, с; **Plant** — объект управления; **RefModel** — эталонная модель.

Модели с обратными связями (LXVI)

Примеры динамических сетей – 1

LDDN – Layered Digital Dynamic Network

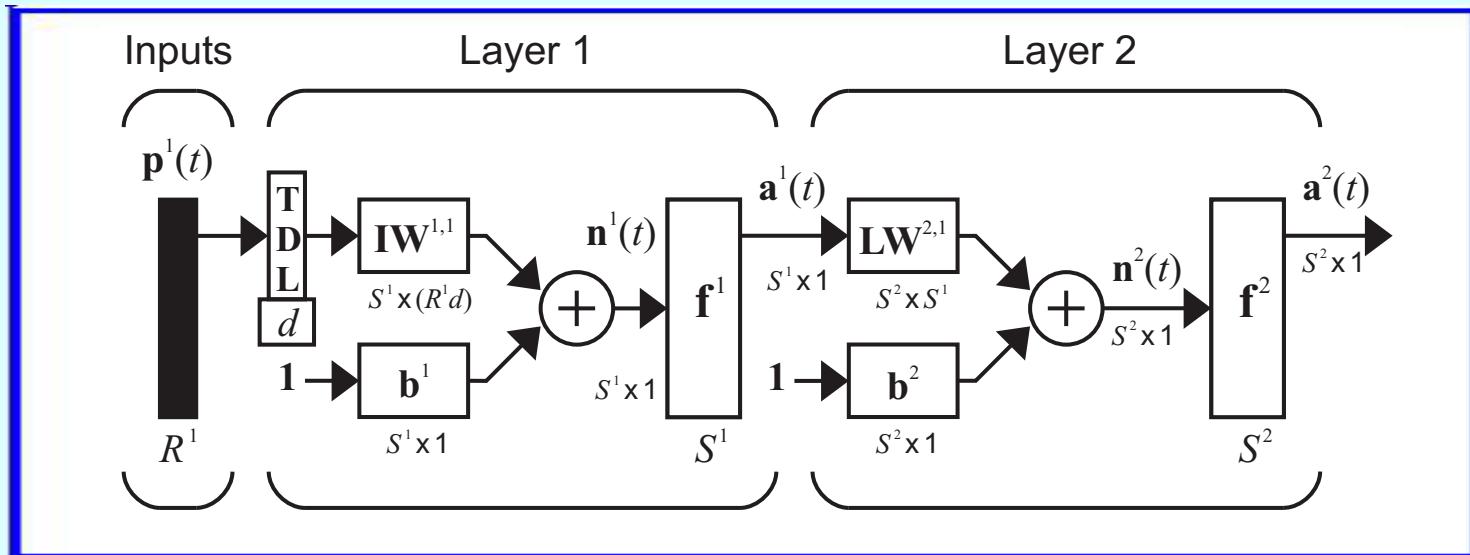


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 6-9).

Динамические модели без обратных связей (I)

Примеры динамических сетей – 2

FTDNN – Focused Time-Delay Neural Network

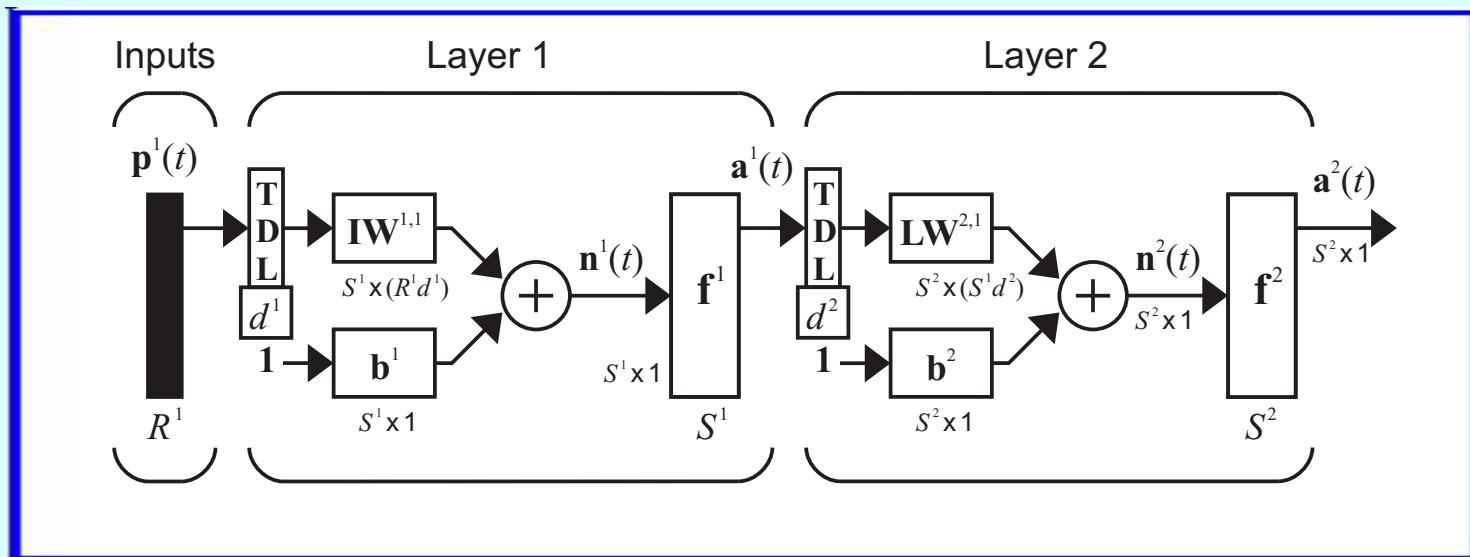


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 6-11).

Динамические модели без обратных связей (II)

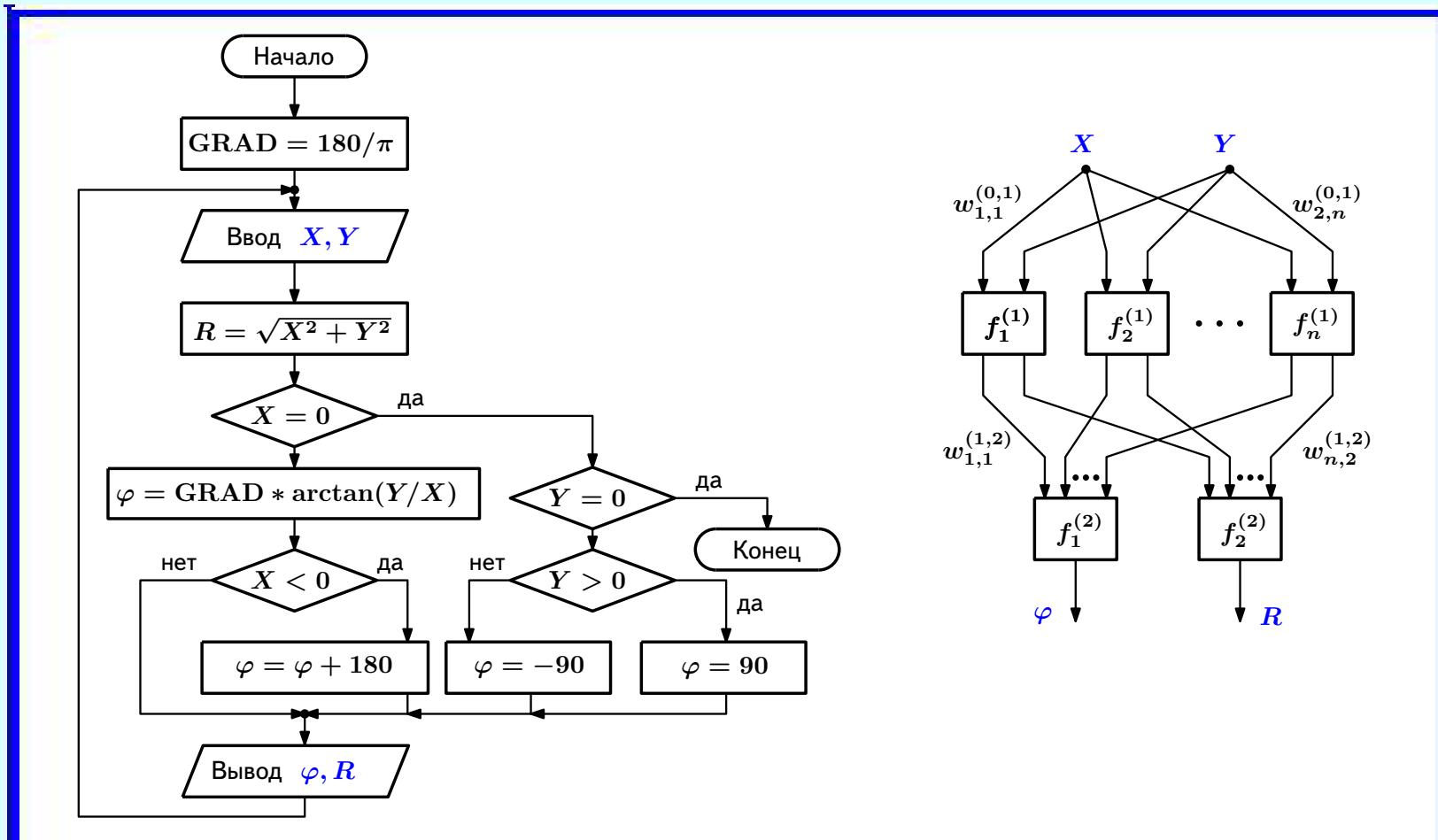
Примеры динамических сетей – 3

DTDNN – Distributed Time-Delay Neural Network



Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 6-15).

Теоретические и эмпирические модели



Можно ли каким-либо образом **объединить**
эти два класса моделей?

Полуэмпирические модели ДС (I)

Общий подход – 1

Общий подход: сочетание **теоретических знаний**, доступных для моделируемой динамической системы (**ДС**), с методами **улучшения** теоретической модели за счет выполнения для нее структурных преобразований и обучения.

Теоретические знания — из первых принципов, из опыта вычислительной математики в моделировании и динамических систем.

Методы улучшения на основе на подходов к формированию и обучению нейросетевых моделей.

Объединение этих двух подходов — динамические модели нейросетевого типа, в архитектуре которых существенным образом учитываются имеющиеся знания об объекте моделирования.

Полуэмпирические модели ДС (II)

Общий подход – 2

«Белый ящик» — теоретические модели

«Черный ящик» — эмпирические модели

«Серый ящик» — полуэмпирические модели

Теоретические модели — формируются путем дедуктивного вывода из первых принципов и следствий из них.

Эмпирические модели — основываются только на экспериментальных данных о поведении системы и не включают в себя никаких элементов, непосредственно отражающих теоретическое знание об объекте.

Полуэмпирические модели — отражают как теоретические знания, так экспериментальные данные об объекте. Эти модели — **динамические сети с модульной архитектурой**.

Полуэмпирические модели ДС (III)

Общий подход – 3

Этапы формирования динамических сетей с модульной архитектурой в виде полуэмпирических НС-моделей:

1. Формирование **теоретической модели** с непрерывным временем для исследуемой динамической системы (ДС), сбор доступных экспериментальных данных о поведении этой системы.
2. **Оценка точности** теоретической модели ДС на доступных данных, в случае недостаточной ее точности выдвижение гипотез о причинах и возможных путях их устранения.
3. Преобразование исходной модели с непрерывным временем в **модель с дискретным временем**.
4. Формирование **нейросетевого представления** для полученной модели с дискретным временем.
5. **Обучение** нейросетевой модели.
6. **Оценка точности** обученной нейросетевой модели.
7. **Корректировка**, в случае недостаточной точности, нейросетевой модели путем внесения в нее структурных изменений.

Полуэмпирические модели ДС (IV)

Формирование и оценка точности теоретической модели – 1

Объект исследования и моделирования — управляемая динамическая система с непрерывным временем.

Теоретическая модель — система обыкновенных дифференциальных уравнений.

Дополнительно требуются **экспериментальные данные** о поведении рассматриваемой системы:

- для оценки точности предлагаемых моделей,
- для подстройки (обучения) модели.

Полуэмпирические модели ДС (V)

Формирование и оценка точности теоретической модели – 2

Эмпирические модели (например, типа NARX и NARMAX) формируются исключительно на основе данных о функционировании моделируемого объекта (в виде последовательности значений входных и выходных сигналов для них).

Полуэмпирическая модель — требуются априорные знания о рассматриваемом объекте.

Эти знания чаще всего имеют форму системы **дифференциальных уравнений**:

$$\begin{aligned}\frac{dx}{dt} &= f(x(t), u(t)), \\ y(t) &= g(x(t)).\end{aligned}\tag{13}$$

Здесь **x** — вектор переменных состояния, **y** — наблюдаемый вектор выходов, **u** — вектор управляющих переменных, **f** и **g** — известные вектор-функции.

Полуэмпирические модели ДС (VI)

Формирование и оценка точности теоретической модели – 3

Модель (13) может быть неудовлетворительной по точности:

$$\begin{aligned}\frac{dx}{dt} &= f(x(t), u(t)), \\ y(t) &= g(x(t)).\end{aligned}$$

Возможные причины:

- функции **f** и **g** известны с недостаточной точностью, например, их описания содержат **параметры**, значения которых известны **недостаточно точно**;
- **недостаточная детализация** модели, т. е. в ней учтены не все значимые факторы вследствие недостаточного уровня теоретического знания о них.

Полуэмпирические модели ДС (VII)

Формирование и оценка точности теоретической модели – 4

**Возможный путь устранения неточности исходной модели –
поэтапное уточнение ее путем:**

- настройки параметров модели,
- структурной корректировки модели,

на основе **экспериментальных данных** о поведении
моделируемой системы.

Полуэмпирические модели ДС (VIII)

Формирование и оценка точности теоретической модели – 5

Инструмент корректировки — нейросетевое обучение.

База корректировки — набор гипотез, каждая из которых представляет собой попытку представить, что именно в модели препятствует ее нормальной работе и каким образом соответствующее препятствие можно устранить.

Структурная корректировка — на модульной основе: объектом ее является некоторая часть модели, заменяемая на другой ее вариант, отвечающий одной из сформулированных гипотез.

Полуэмпирические модели ДС (IX)

Формирование и оценка точности теоретической модели – 6

Модельный пример динамической системы (ДС):

$$\begin{aligned}\dot{x}_1(t) &= -(x_1(t) + 2x_2(t))^2 + u(t) \\ \dot{x}_2(t) &= 8.322109 \sin(x_1(t)) + 1.135x_2(t)\end{aligned}\quad (14)$$

Для этой ДС: временной интервал $[0, T]$ с шагом дискретизации Δt ;
начальные условия $x_i(0)$.

Вектор состояния $(x_1(t), x_2(t))$ — частично наблюдаемый:
 $y(t) = x_2(t)$, с аддитивным белым шумом, действующим на выход, с
заданным среднеквадратичным отклонением σ .

Полуэмпирические модели ДС (Х)

Формирование и оценка точности теоретической модели – 7

Логика структурной корректировки модели

Первое уравнение из (14) — в неизменном виде (будем считать, что знание об объекте, выражаемое этим уравнением, совершенно точное), **второе** — вначале в упрощенном виде.

Стартовый вариант модельного примера ДС:

$$\dot{x}_1(t) = -(x_1(t) + 2x_2(t))^2 + u(t)$$

$$\dot{x}_2(t) = \textcolor{red}{8.32}x_1(t)$$

Далее — **последовательное усложнение** для повышения точности моделирования.

Полуэмпирические модели ДС (XI)

Дискретизация теоретической модели с непрерывным временем – 1

Корректировка модели для компенсации пробелов в теоретических знаниях об объекте — **на основе экспериментальных данных** о его поведении.

Эффективный **инструмент корректировки** — методы обучения НС-моделей.

Для использования этого инструмента **теоретическую модель** необходимо **преобразовать в НС-форму**.

Первый шаг — переход от исходной модели с непрерывным временем, т. е. от системы обыкновенных **дифференциальных уравнений**, к модели динамической системы с дискретным временем, т. е. к системе **разностных уравнений**.

Полуэмпирические модели ДС (XII)

Дискретизация теоретической модели с непрерывным временем – 2

Возможность привлечь к формированию НС-модели **еще один вид знаний** (помимо знаний «физического» характера о природе моделируемого объекта): знания **относительно эффективности использования различных разностных схем для решения тех или иных классов дифференциальных уравнений**.

Учет такого знания дает возможность уйти от концепции «черного ящика» **при выполнении дискретизации**.

В отличие от авторегрессионных моделей типа NARX или NARMAX, это позволяет выбирать схему дискретизации из числа тех, которые показывают **наибольшую эффективность** на уравнениях именно такого класса, к которому относятся уравнения полученной теоретической модели.

Полуэмпирические модели ДС (XIII)

Дискретизация теоретической модели с непрерывным временем – 3

Алгоритмическая база для дискретизации моделей с непрерывным времени — **численные методы** решения обыкновенных дифференциальных уравнений.

Два класса методов:

- одношаговые методы**, использующие единственное стартовое значение;
- многошаговые методы**, использующие совокупность из нескольких стартовых значений.

Оба этих класса методов могут быть использованы в качестве основы для формирования требуемых моделей с дискретным временем.

Полуэмпирические модели ДС (XIV)

Дискретизация теоретической модели с непрерывным временем – 4

Для перехода **к дискретному времени** в модельной задаче — **одношаговая схема Эйлера** 1-го порядка и **многошаговая схема Адамса** 4-го порядка точности (начальные значения рассчитываются с помощью схемы Рунге-Кутты 4-го порядка).

Схема Эйлера:

$$X(i+1) = X(i) + TF(i). \quad (15)$$

Схема Адамса:

$$X(i+1) = X(i) + \frac{T}{24} [55F(i) - 59F(i-1) + 37F(i-2) - 9F(i-3)]. \quad (16)$$

Здесь $X(i) \equiv X(ih)$, $i = 0, 1, \dots, \lfloor T/h \rfloor$ — вектор состояний в момент времени $t = ih$.

Полуэмпирические модели ДС (XV)

Нейросетевое представление модели с дискретным временем – 5

Шаг, следующий за дискретизацией теоретической модели — **преобразование** полученной разностной модели в **нейросетевую форму**.

С этой целью величины и связи между ними в разностной модели **интерпретируются** в терминах элементов нейросетевых моделей.

В результате получается **рекуррентная нейронная сеть**, взаимно однозначно соответствующая интерпретируемой разностной модели.

Полуэмпирические модели ДС (XVI)

Нейросетевое представление модели с дискретным временем – 1

различные модели с непрерывным временем



различные разностные модели



различные рекуррентные сети

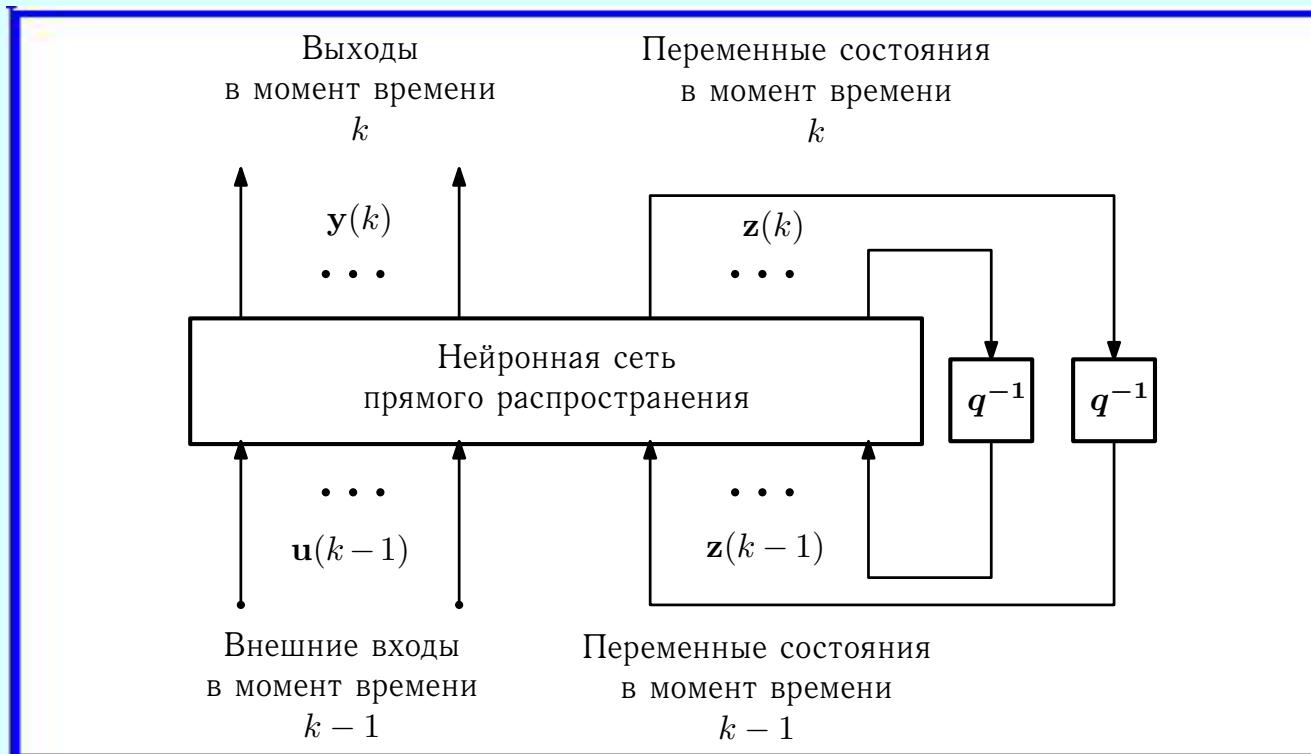
Топология связей в полученной рекуррентной НС-модели в общем случае достаточно **сложная и разнообразная**.

Непосредственное решение задач обучения для таких сетей требует **различных вариантов** методов обучения, согласованных с архитектурой полученных рекуррентных сетей.

Полуэмпирические модели ДС (XVII)

Нейросетевое представление модели с дискретным временем – 2

Унификация подхода к их обучению — приведение получаемых рекуррентных НС-моделей к **однообразному виду**.

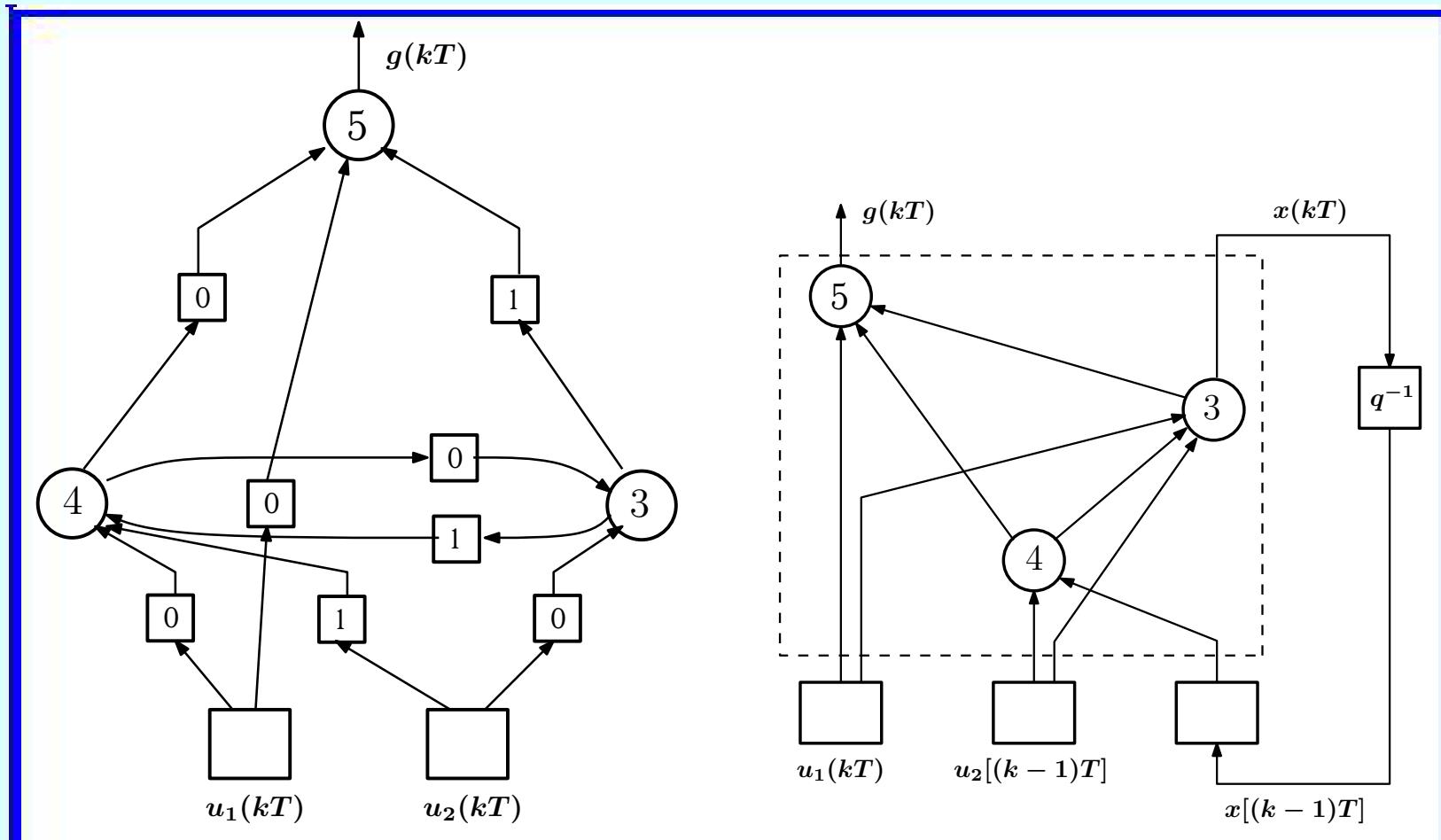


Результат преобразования — **НС-модель канонического вида**, представляющая собой **слоистую сеть прямого распространения**, замкнутую внешними обратными связями с единичными задержками от выходов к входам этой сети.

Полуэмпирические модели ДС (XVIII)

Нейросетевое представление модели
с дискретным временем – 3

Пример 1. Исходная рекуррентная НС и ее каноническое представление



Полуэмпирические модели ДС (XIX)

Нейросетевое представление модели с дискретным временем – 4

Пример 2. Объект описывается следующей системой уравнений:

$$\begin{aligned}\ddot{x}_1 &= \phi_1(x_1, x_2, x_3, u), \\ x_2 &= \phi_2(x_1, x_3), \\ \ddot{x}_3 &= \phi_3(x_1, \dot{x}_2), \\ y &= x_3.\end{aligned}\tag{17}$$

Явный метод Эйлера: основан на аппроксимации следующего вида для производной по времени функции $f(\cdot)$ в момент времени kT :

$$\{f[(k+1)T] - f(kT)\}/T,$$

где T – период дискретизации (шаг интегрирования) для исходной системы (17).

Полуэмпирические модели ДС (ХХ)

Нейросетевое представление модели с дискретным временем – 5

Пример 2 (прод.) Представление системы (17) **в форме с дискретным временем**, основанное на использовании явного метода Эйлера:

$$\begin{aligned}x_1(k+1) &= \psi_1[x_1(k), x_1(k-1), x_2(k-1), x_3(k-1), u(k-1)], \\x_2(k+1) &= \psi_2[x_1(k+1), x_3(k+1)], \\x_3(k+1) &= \psi_3[x_3(k), x_3(k-1), x_1(k-1), x_2(k), x_2(k-1)], \\y(k+1) &= x_3(k+1).\end{aligned}\tag{18}$$

Полуэмпирические модели ДС (ХХI)

Нейросетевое представление модели с дискретным временем – 6

Пример 2 (прод.) Для перехода к канонической форме следует ввести **новый вектор переменных состояния**:

$$\mathbf{z}(k) = [x_1(k), x_2(k - 1), x_3(k), x_3(k - 1)]^T.$$

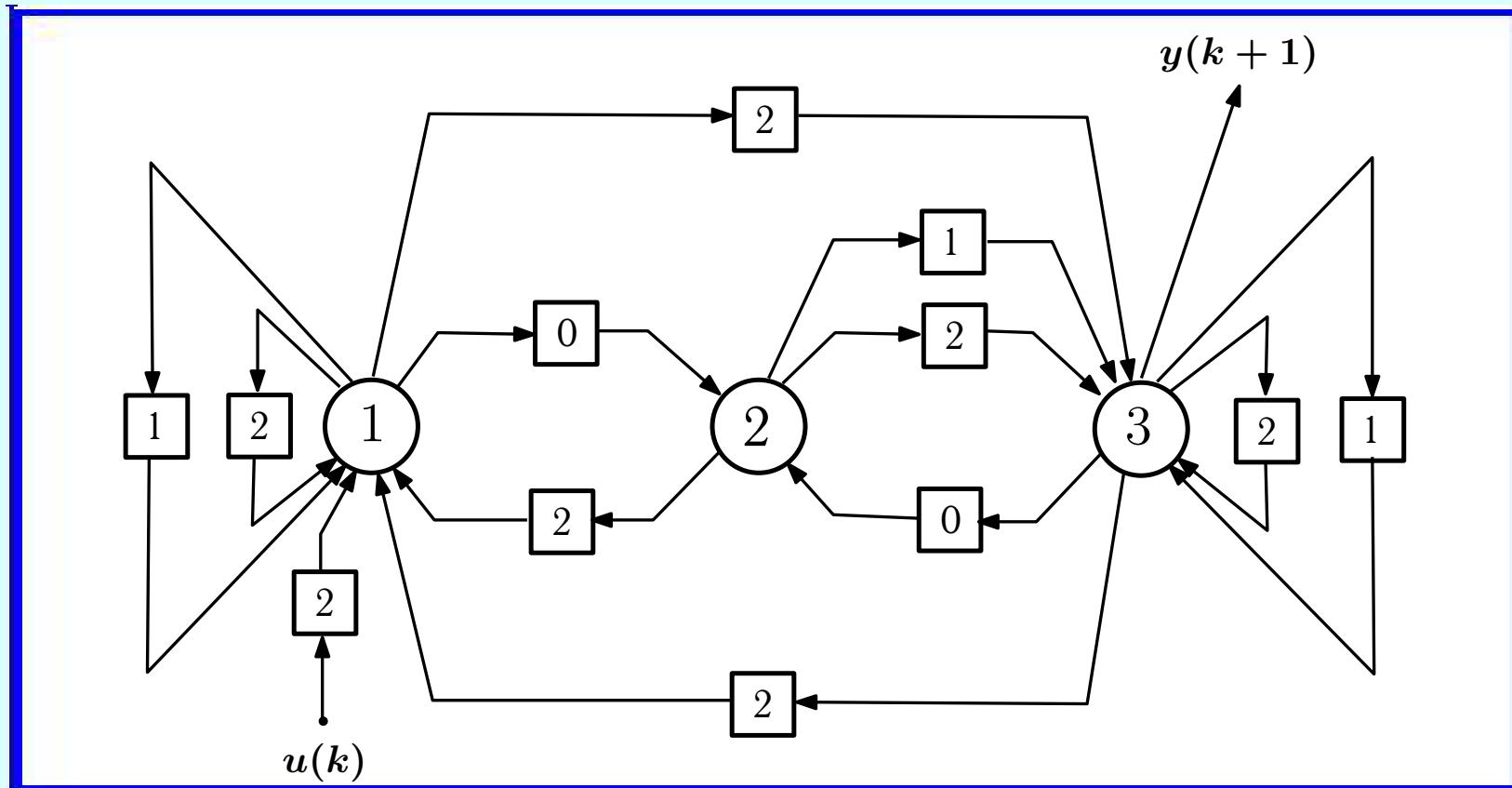
Каноническая форма рекуррентной нейронной сети при таком выборе вектора состояния представляет собой **слоистую сеть прямого распространения** с элементами единичных задержек в ней и с обратными связями, внешними по отношению к этой сети.

Исходная рекуррентная сеть и ее каноническая форма **полностью эквивалентны**.

Полуэмпирические модели ДС (ХХII)

Нейросетевое представление модели
с дискретным временем – 7

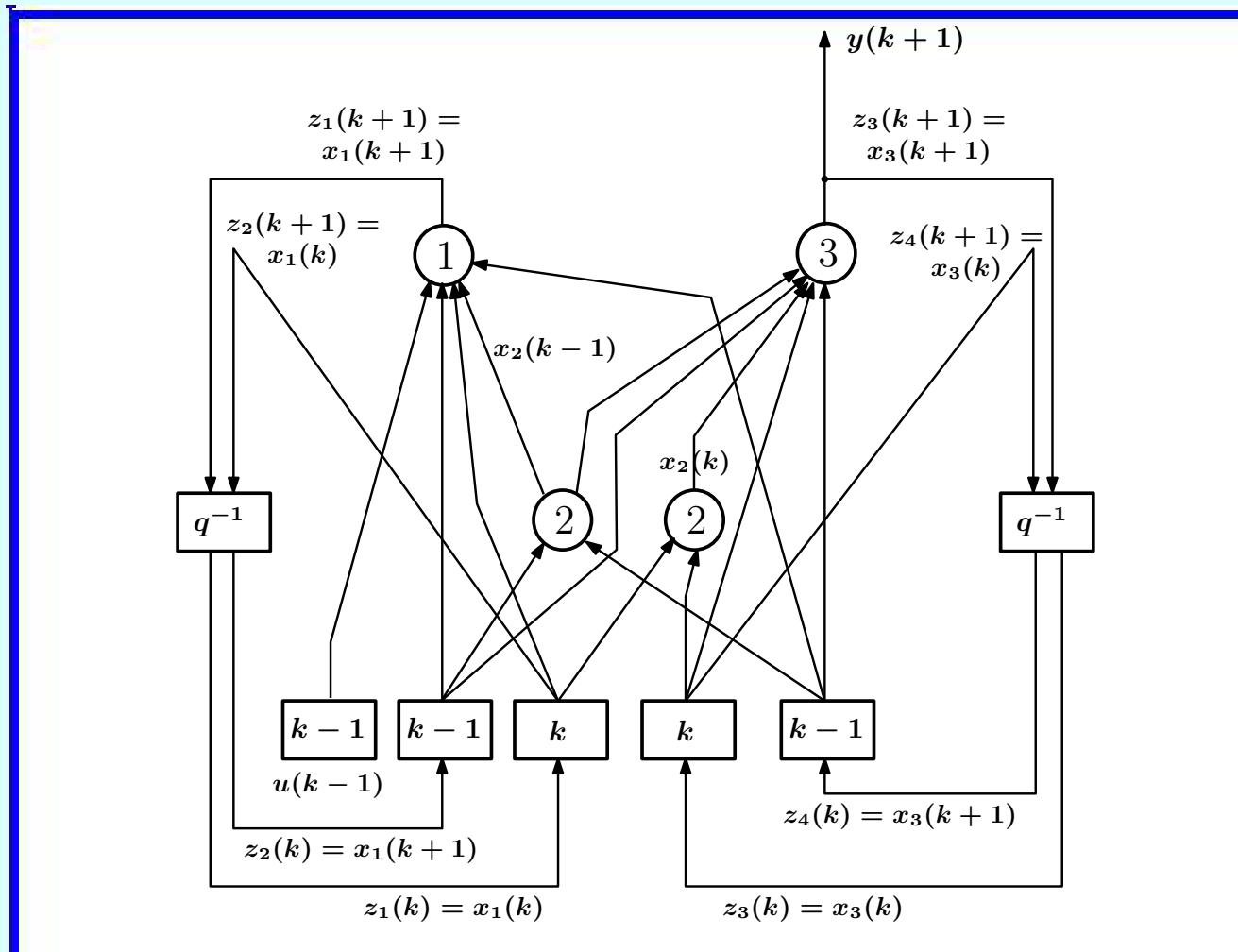
Исходная рекуррентная НС (пример 2)



Полуэмпирические модели ДС (ХХIII)

Нейросетевое представление модели
с дискретным временем – 8

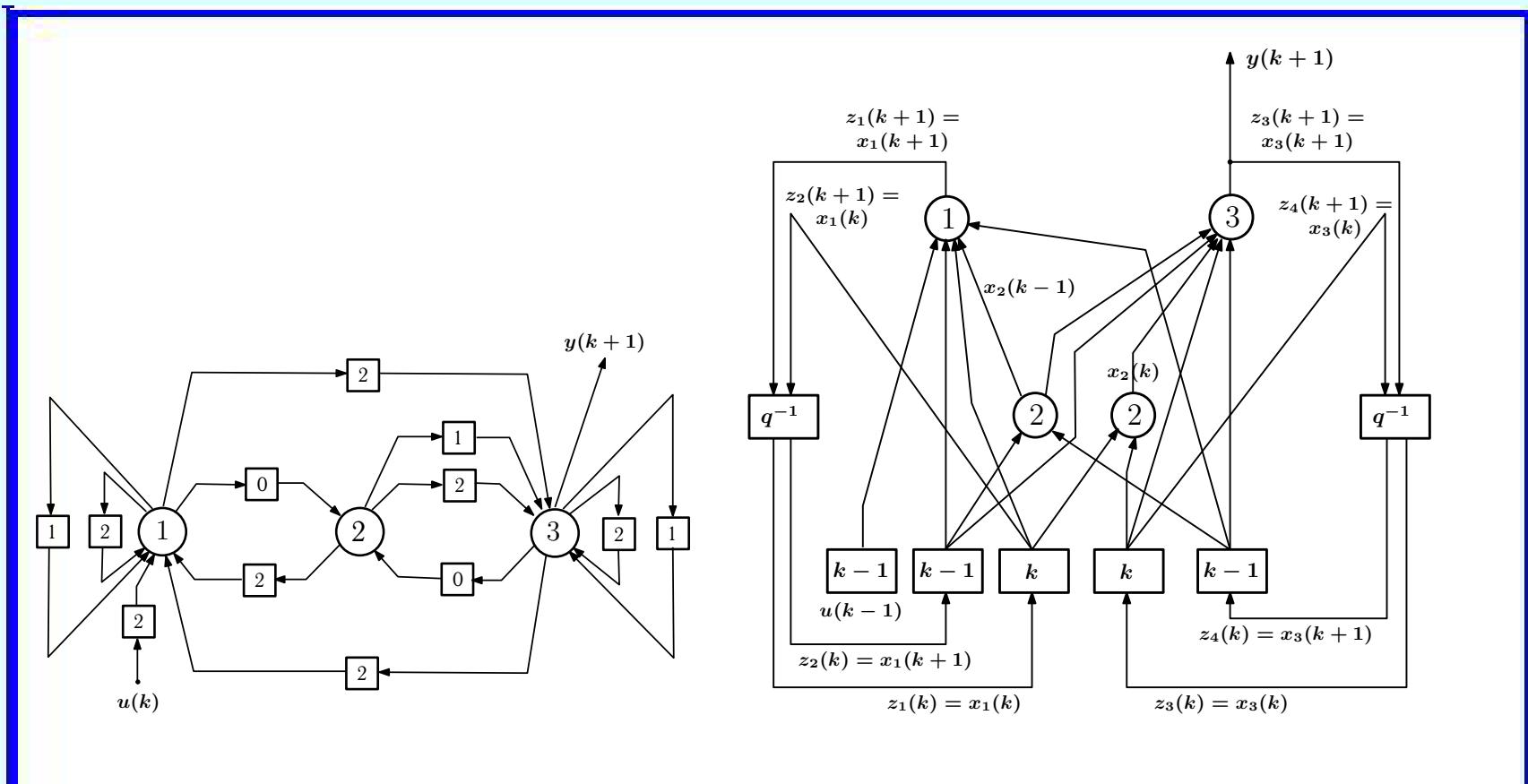
Каноническое представление НС (пример 2)



Полуэмпирические модели ДС (ХIV)

Нейросетевое представление модели
с дискретным временем – 9

Исходная НС и ее каноническое представление (пример 2)



Полуэмпирические модели ДС (ХV)

Нейросетевое представление модели
с дискретным временем – 10

В рассматриваемой **модельной задаче**

$$\dot{x}_1(t) = -(x_1(t) + 2x_2(t))^2 + u(t)$$

$$\dot{x}_2(t) = 8.322109 \sin(x_1(t)) + 1.135x_2(t)$$

используемые схемы дискретизации позволяют получить каноническое представление сети либо сразу (**для схемы Эйлера (15)**), либо после незначительной корректировки исходного неканонического варианта (**для схемы Адамса (16)**).

В более сложных случаях такое преобразование провести потребуется.

Полуэмпирические модели ДС (ХХVI)

Нейросетевое представление модели
с дискретным временем – 11

Стартовый вариант модельной задачи:

$$\dot{x}_1(t) = -(x_1(t) + 2x_2(t))^2 + u(t)$$

$$\dot{x}_2(t) = \mathbf{8.32}x_1(t)$$

Полуэмпирические модели ДС (ХVII)

Нейросетевое представление модели с дискретным временем – 12

Простейший метод дискретизации – **явный метод Эйлера**, в котором производная $df(t)/dt$ аппроксимируется разностным соотношением вида

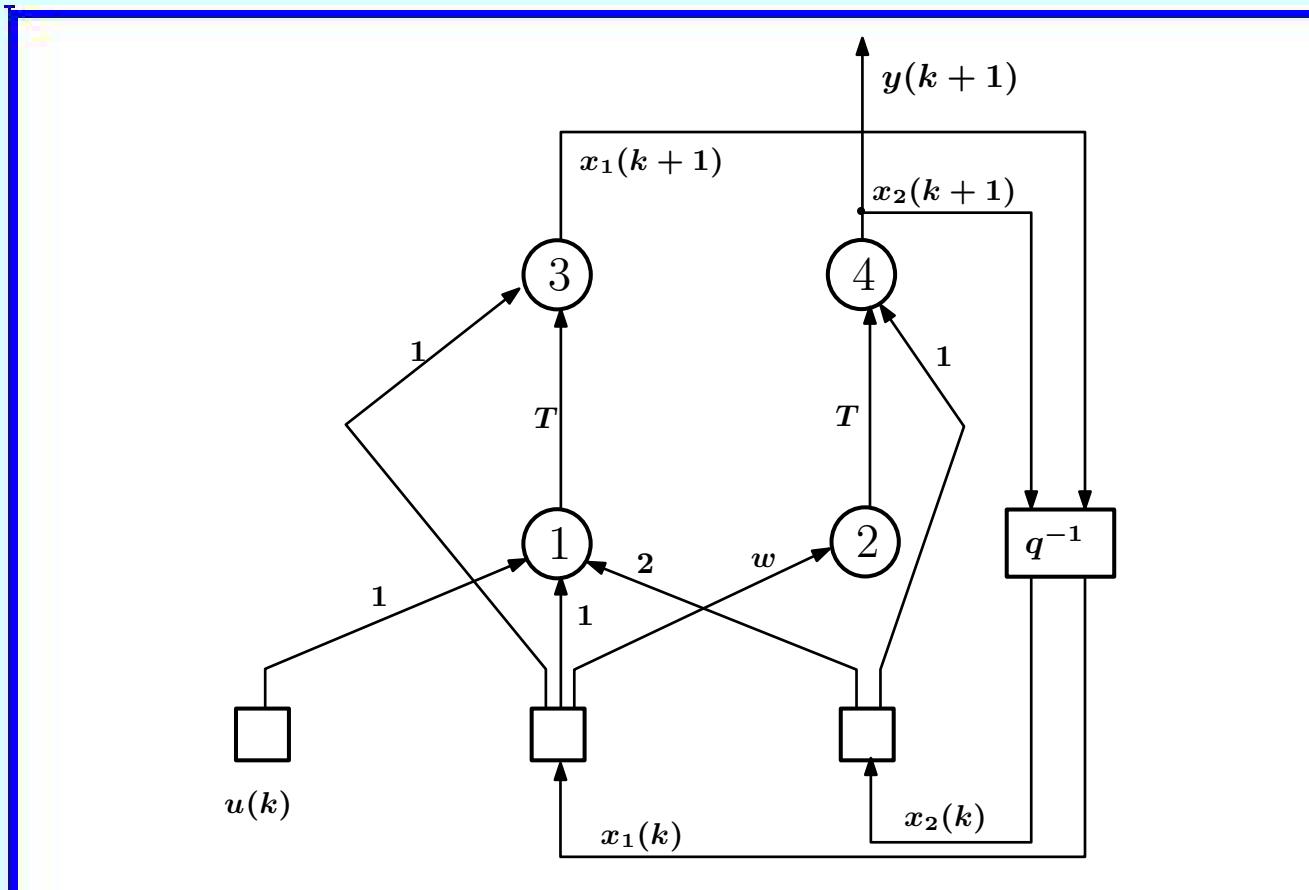
$$\frac{df(t)}{dt} \approx \frac{f[(k+1)T] - f(kT)}{T},$$

где k – положительное целое число. Эта схема дискретизации дает следующую **модель с дискретным временем**:

$$x_1[(k+1)T] = x_1(kT) + T[-(x_1(kT) + 2x_2(kT))^2 + u(kT)],$$
$$x_2[(k+1)T] = x_2(kT) + T(8.32x_1(kT))$$

Полуэмпирические модели ДС (ХХVIII)

Нейросетевое представление модели с дискретным временем – 13



Каноническая форма исходной теоретической модели,
дискретизированной с использованием явного метода Эйлера

Полуэмпирические модели ДС (XXIX)

Нейросетевое представление модели
с дискретным временем – 14

Упрощающие предположения:

- постоянные входы (**смещения**) нейронов сети не показаны,
- дискретные **моменты времени** kT обозначаются как k ,
- обозначение q^{-1} соответствует элементу **единичной задержки**.

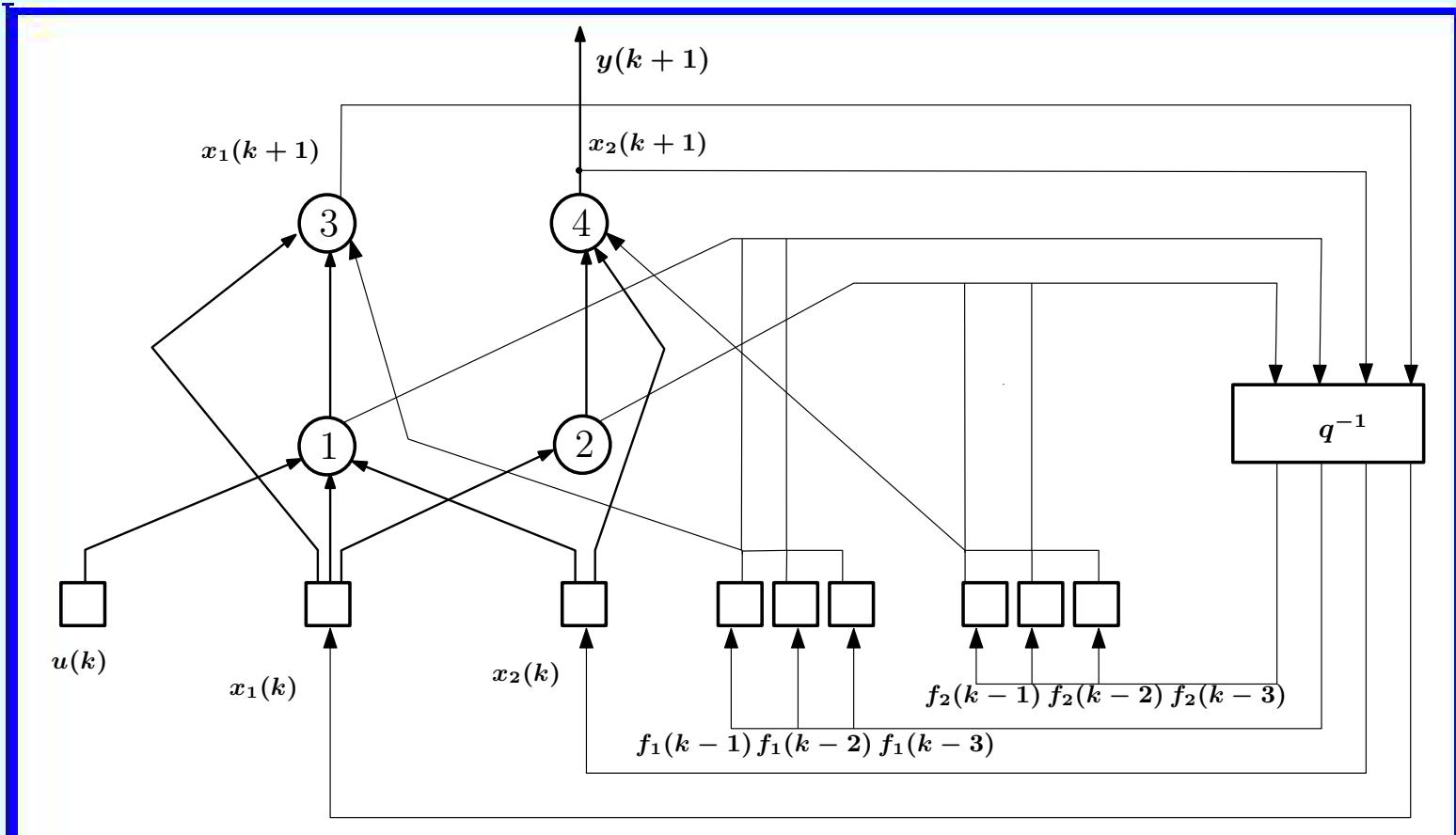
Нейрон 1 формирует взвешенную сумму сигналов x_1 и x_2 , с весами, показанными непосредственно на рисунке, за которой следует умножение на нелинейность $-s^2$, а также добавляет к полученному результату значение управляющего сигнала $u(k)$.

Нейрон 2 умножает свой входной сигнал на весовой коэффициент w .

Нейроны 3 и 4 реализуют операции взвешенного суммирования своих входов.

Полуэмпирические модели ДС (XXX)

Нейросетевое представление модели с дискретным временем – 15



Каноническая форма исходной теоретической модели,
дискретизированной с использованием явного метода Адамса

Полуэмпирические модели ДС (ХХI)

Нейросетевое представление модели

с дискретным временем – 16

Полученное **нейросетевое представление** формируемой модели, выраженное в каноническом виде, **обучается** с помощью соответствующих НС-алгоритмов.

Характерная особенность модели: Алгоритм перехода от разностного к сетевому представлению модели позволяет **сохранить в явном виде** в нейросетевой форме **локализацию выполняемых функций**, которая имела место в исходной модели с непрерывным временем и, соответственно, в отвечающей ей разностной модели.

Это дает возможность часть сформированной НС-модели зафиксировать и **не подвергать изменениям** (как структуры, так и параметров этой части) в процессе обучения.

Такой подход оправдан для фрагментов модели, основанных **на точном теоретическом знании**, т. е. на таком знании, которое в рамках решаемой прикладной задачи не вызывает сомнений.

Полуэмпирические модели ДС (XXXII)

Нейросетевое представление модели с дискретным временем – 17

Те части модели, которые **вызывают сомнения** (неточности в них могут считаться причиной неудовлетворительных характеристик модели), **подлежат корректировке**, как параметрической, так и структурной, выполняемой на модульной основе.

Таким образом, **в процессе обучения** для НС-модели полуэмпирического типа («серый ящик») **настройке** (обучению) подвергается, как правило, **только часть модели** при фиксированной структуре настраиваемой части, тогда как остальные ее фрагменты остаются неизменными, **сохраняющие структуру связей и значения параметров** такими, какие были получены при переводе теоретической модели из разностной в сетевую форму.

Полуэмпирические модели ДС (XXXIII)

Структурная корректировка полуэмпирической модели – 1

Если **при фиксированной структуре** настраиваемой (обучаемой) части НС-модели **не удается получить** от модели приемлемых характеристик **только за счет обучения** — в рассматриваемом примере это **подбор** значения коэффициента в уравнении

$$\dot{x}_2(t) = 8.32x_1(t),$$

это означает, что в настраиваемой части **необходимы структурные изменения.**

Такие изменения осуществляются **на основе ранее сформулированных гипотез**, представляющих собой соображения о возможных причинах неудовлетворительного поведения модели, а также о путях устранения этих причин.

Полуэмпирические модели ДС (XXXIV)

Структурная корректировка полуэмпирической модели – 2

Варианты структурной корректировки:

$$\dot{x}_2(t) = 8.32x_1(t)$$



$$\dot{x}_2(t) = \varphi(x_1(t))$$



$$\dot{x}_2(t) = \Psi(x_1(t), x_2(t))$$

Здесь $\varphi(x_1(t))$ и $\Psi(x_1(t), x_2(t))$ – нелинейные функции.

Все корректировки локальны, они непосредственно затрагивают только часть формируемой НС-модели, не меняя остальных ее частей.

Такого рода **модульность** и **пошаговый процесс** внесения структурных изменений (путем последовательного применения предлагаемых гипотез) в модель существенно облегчают формирование НС-модели, обладающей требуемыми характеристиками.

Полуэмпирические модели ДС (XXXV)

Экспериментальная оценка полуэмпирической модели – 1

Исходная система – введенная ранее модельная задача:

$$\begin{aligned}\dot{x}_1(t) &= -(x_1(t) + 2x_2(t))^2 + u(t), \\ \dot{x}_2(t) &= 8.322109 \sin(x_1(t)) + 1.135x_2(t),\end{aligned}$$

рассматриваемая на временном интервале $t \in [0; 100]$ с шагом дискретизации $\Delta t = 0.025$ с и начальными условиями $x_1(0) = x_2(0) = 0$.

Вектор состояния является **частично наблюдаемым**: $y(t) = x_2(t)$, с **аддитивным белым гауссовским шумом** со среднеквадратичным отклонением (СКО) $\sigma = 0.01$, действующим **на выход** системы $y(t)$.

Полуэмпирические модели ДС (XXXVI)

Экспериментальная оценка полуэмпирической модели – 2

В идеальном варианте, когда НС-модель **абсолютно точно** воспроизводит исходную систему дифференциальных уравнений, **ошибка моделирования** будет полностью определяться шумом, действующим на выход системы.

Из этого следует, что сопоставление ошибки моделирования с СКО шума позволяет судить о том, **насколько успешно решена задача** моделирования рассматриваемой системы, а заданное значение СКО шума можно принять за **целевое значение ошибки моделирования**.

Полуэмпирические модели ДС (XXXVII)

Экспериментальная оценка полуэмпирической модели – 3

Оптимизация параметров сети: **алгоритм Левенберга-Марквардта**, вычисление матрицы Якоби осуществляется по алгоритму **RTRL** (Real-Time Recurrent Learning, который также известен как алгоритм Forward Propagation, Forward Perturbation).

Критерий оптимизации — среднеквадратичная ошибка предсказания модели на обучающей выборке:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (t_i - y_i)^2}$$

Здесь N — количество элементов обучающей выборки,
 $t = \{t_1, \dots, t_N\}$ — вектор целевых значений наблюдаемой переменной,
 $y = \{y_1, \dots, y_N\}$ — вектор выходов нейросетевой модели.

Полуэмпирические модели ДС (XXXVIII)

Экспериментальная оценка полуэмпирической модели – 4

Пусть **на первом шаге** проводимого вычислительного эксперимента требуемая теоретическая модель не определена полностью: точно известен вид первого уравнения, **вид второго уравнения не известен**.

Примем в качестве **первой рабочей гипотезы**, что требуемое второе уравнение можно записать так, как показано ниже:

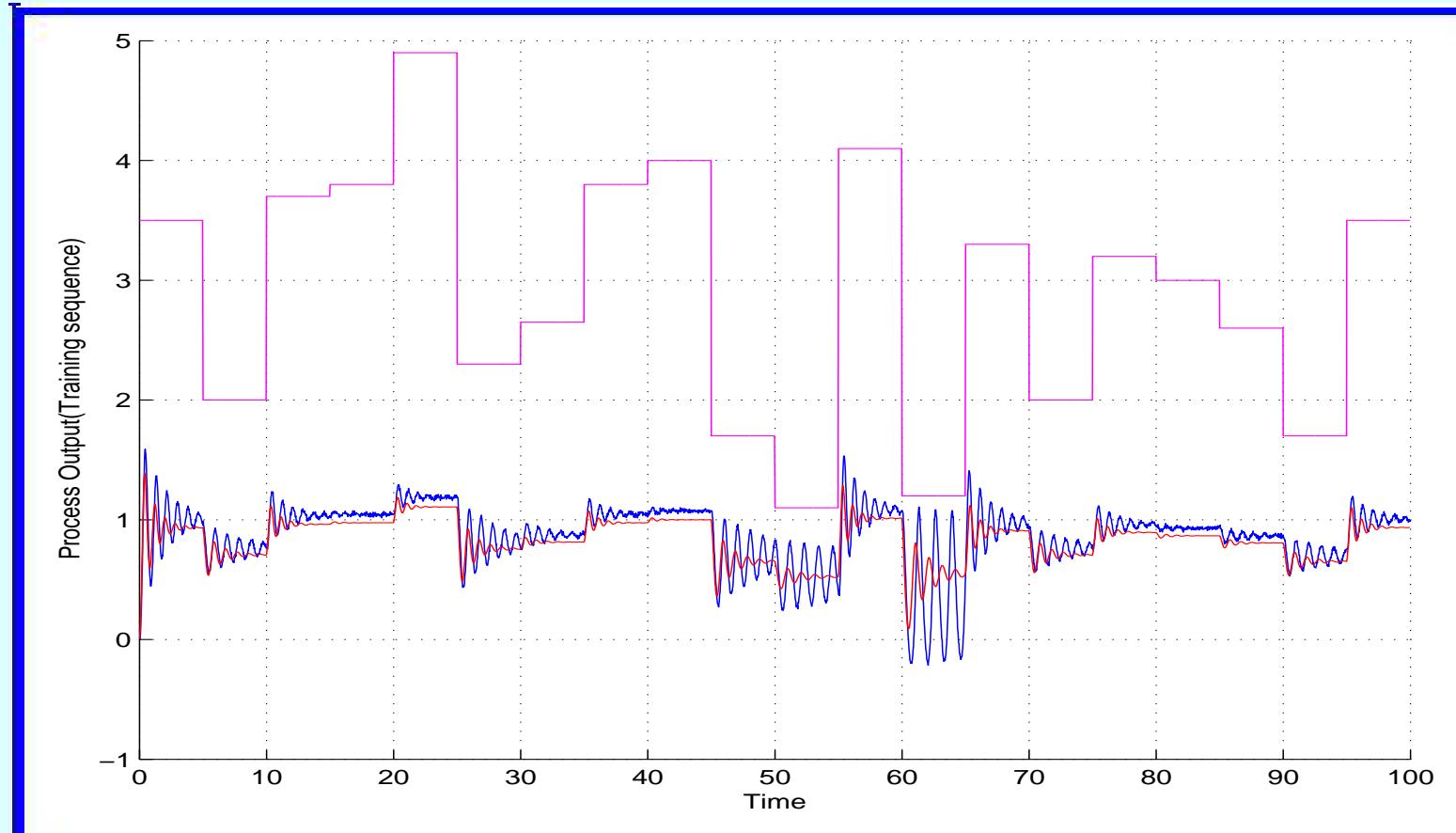
$$\begin{aligned}\dot{x}_1(t) &= -(x_1(t) + 2x_2(t))^2 + u(t) \\ \dot{x}_2(t) &= 8.32x_1(t)\end{aligned}\tag{19}$$

Если эту модель использовать **для анализа поведения** рассматриваемой системы, привлекая методы численного интегрирования систем обыкновенных дифференциальных уравнений, то результат оказывается **неудовлетворительным**.

СКО, получаемая при таком моделировании, равняется **0.1347** при использовании **метода Эйлера** и **0.0434** при использовании **метода Адамса**, что **многократно превышает** целевое значение **0.01**.

Полуэмпирические модели ДС (XXXIX)

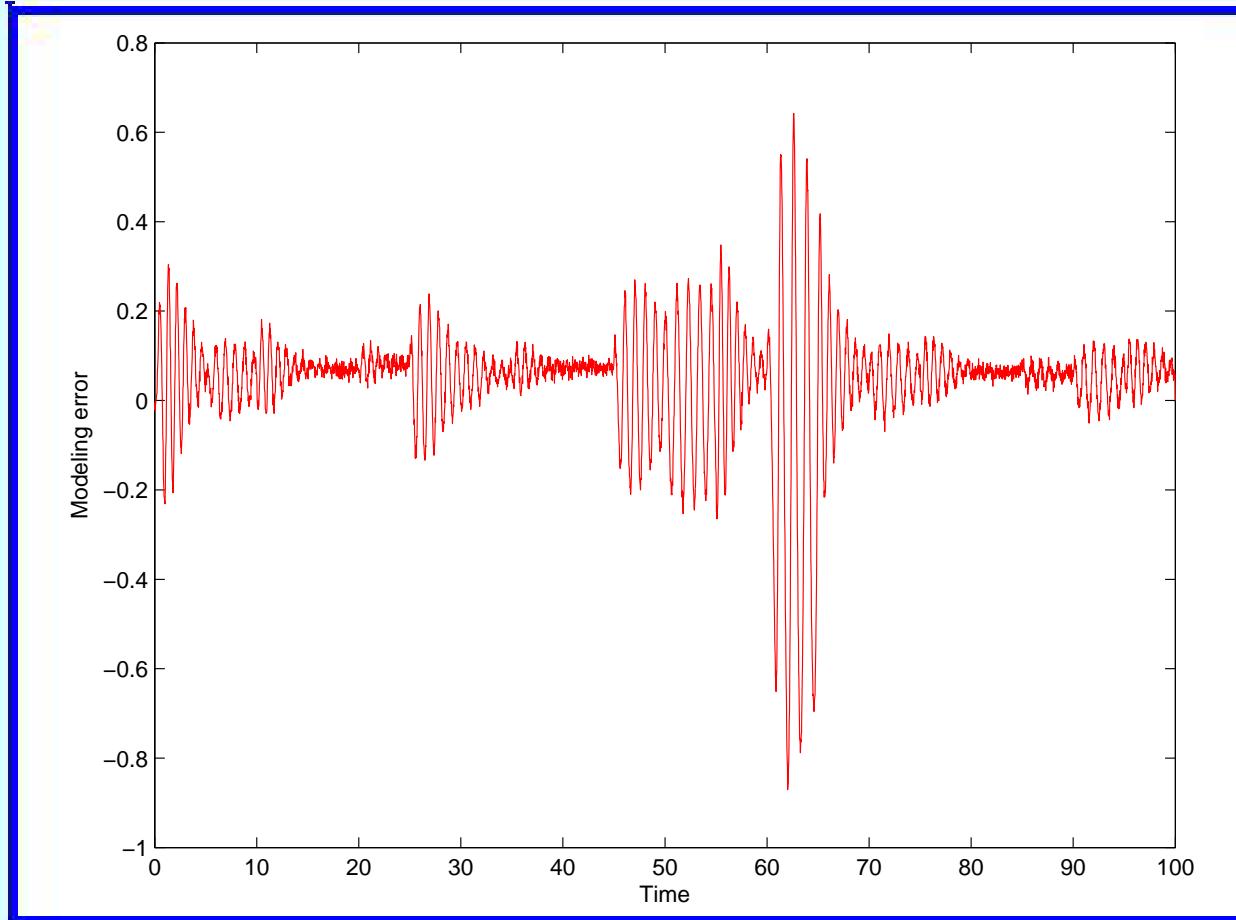
Экспериментальная оценка полуэмпирической модели – 5



Результаты вычислительного эксперимента для модели, полученной согласно первой гипотезе относительно вида второго уравнения, при использовании метода Эйлера: управляемый сигнал $u(t)$ (верхний график); сопоставление ошибки выхода (синяя кривая) и шума измерений (красная кривая)

Полуэмпирические модели ДС (XL)

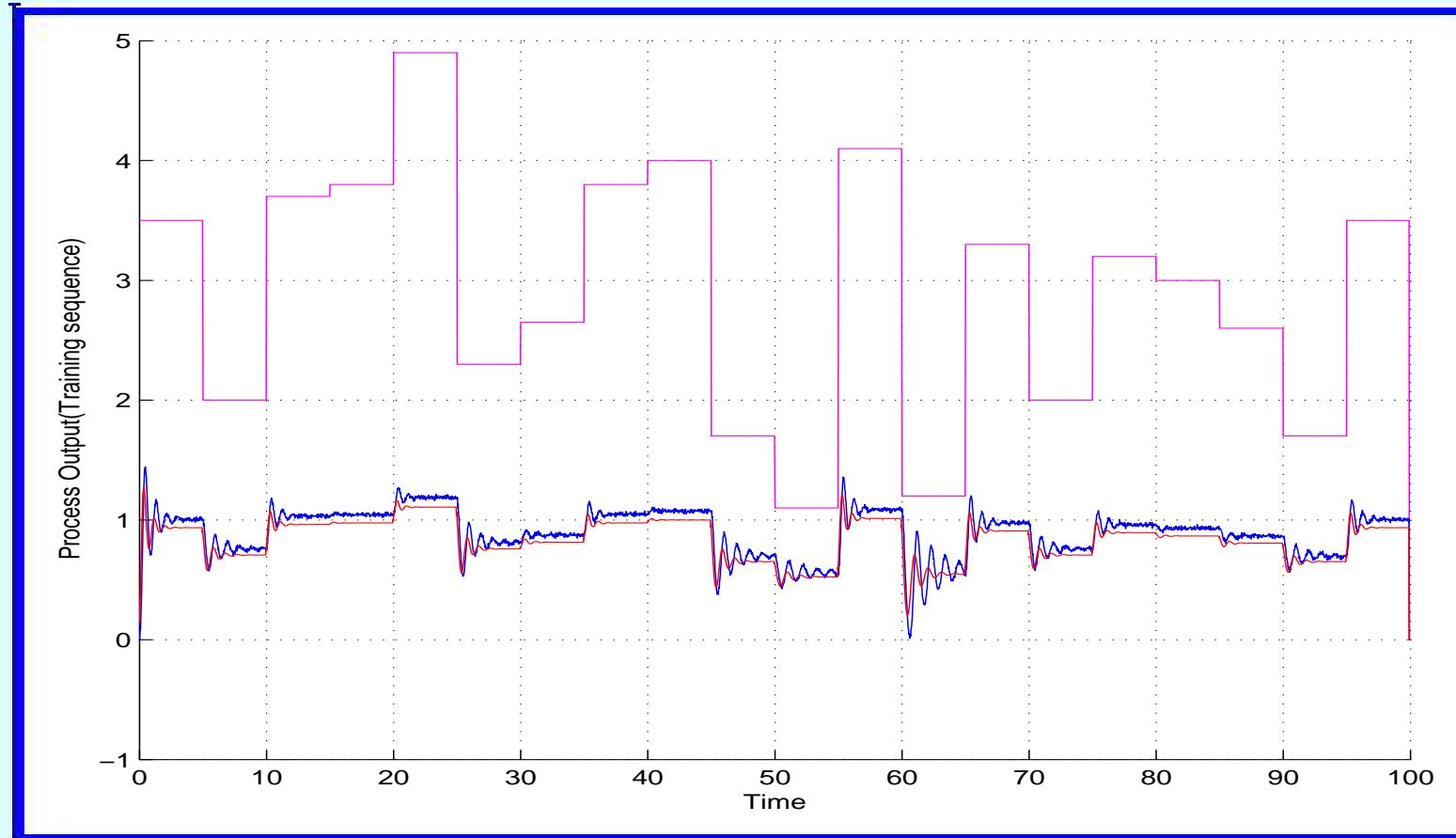
Экспериментальная оценка полуэмпирической модели – 6



Ошибка моделирования для модели, полученной согласно первой гипотезе относительно вида второго уравнения, при использовании метода Эйлера

Полуэмпирические модели ДС (ХЛІ)

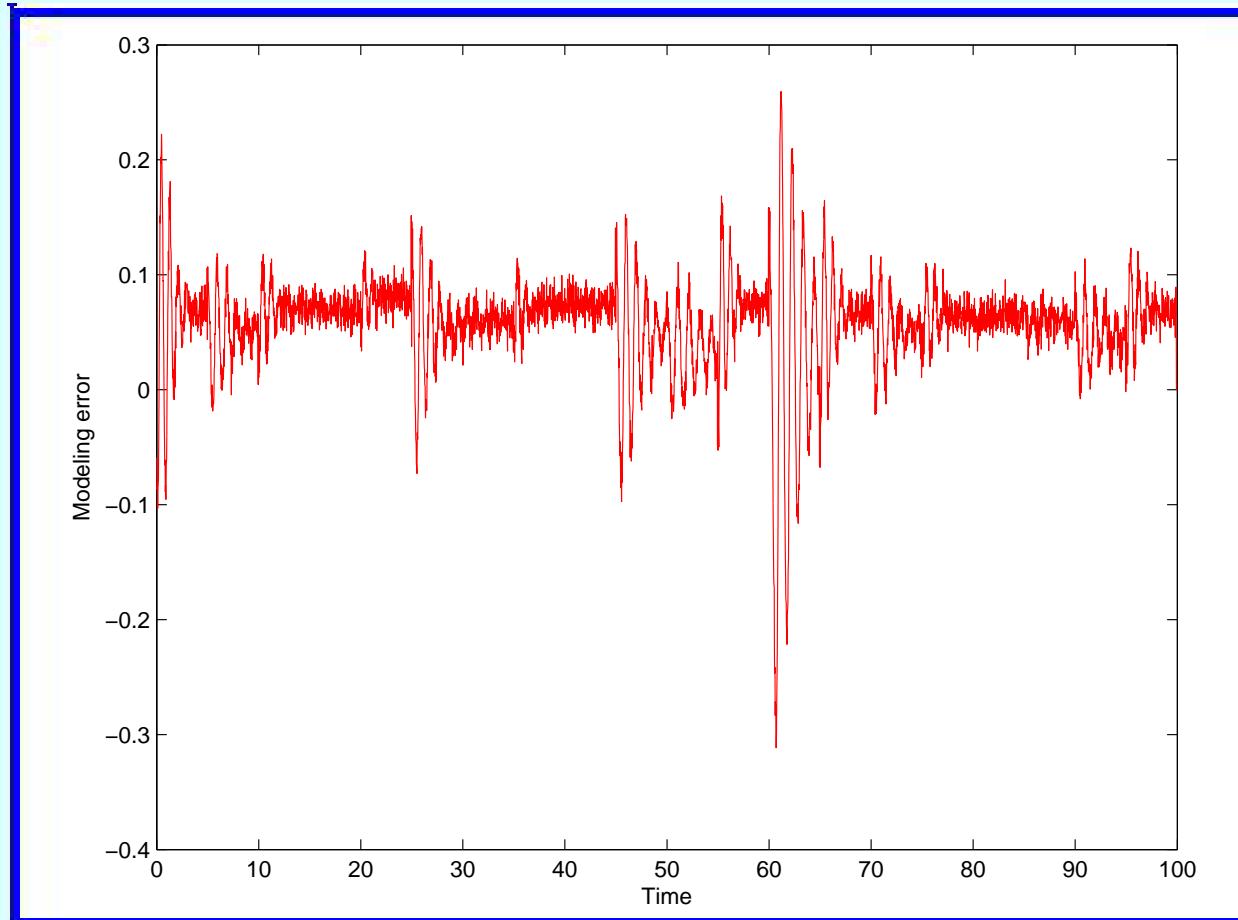
Экспериментальная оценка полуэмпирической модели – 7



Результаты вычислительного эксперимента для модели, полученной согласно первой гипотезе относительно вида второго уравнения, при использовании метода Адамса: управляемый сигнал $u(t)$ (верхний график); сопоставление ошибки выхода (синяя кривая) и шума измерений (красная кривая)

Полуэмпирические модели ДС (XLII)

Экспериментальная оценка полуэмпирической модели – 8



Ошибка моделирования для модели, полученной согласно первой гипотезе
относительно вида второго уравнения, при использовании метода Адамса

Полуэмпирические модели ДС (XLIII)

Экспериментальная оценка полуэмпирической модели – 9

Первое уравнение — точное.

Причина неудачи обусловлена **вторым уравнением**, именно оно не позволяет получить требуемую точность моделирования.

Возможные причины состоят в следующем:

- значение **8.32 числового параметра** в этом уравнении **может быть неточным**, т. е. следует **попытаться подобрать** другое значение данного параметра;
- **вызывает сомнения линейный характер зависимости**, используемый в правой части данного уравнения, т. е. если корректировка значения числового параметра из предыдущего пункта не позволит получить решение с требуемой точностью, необходимо **попытаться использовать** какое-либо **нелинейное соотношение** в правой части второго уравнения;
- отсутствие переменной x_2 в правой части второго уравнения также может послужить **причиной недостаточной точности** модели, т. е. если меры, предложенные в двух предыдущих пунктах, не дадут требуемого результата, следует попытаться **переформулировать правую часть** уравнения с привлечением x_2 .

Полуэмпирические модели ДС (XLIV)

Экспериментальная оценка полуэмпирической модели – 10

Полуэмпирическая форма представления модели дает возможность внести в нее требуемые изменения.

Это осуществляется путем введения **модуля-подсети**, реализующего **необходимую нелинейность**, в НС-модель, полученную дискретизацией из соответствующей теоретической модели.

При этом, если **изменения** вносить в том порядке, в котором они были перечислены, сложность модели будет последовательно возрастать и можно ожидать, что **точность ее также будет возрастать**.

Полуэмпирические модели ДС (ХLV)

Экспериментальная оценка полуэмпирической модели – 11

Обучаемый вариант модельной задачи:

$$\dot{x}_1(t) = -(x_1(t) + 2x_2(t))^2 + u(t)$$

$$\dot{x}_2(t) = \textcolor{red}{8.32}x_1(t)$$



$$\dot{x}_1(t) = -(x_1(t) + 2x_2(t))^2 + u(t)$$

$$\dot{x}_2(t) = \textcolor{red}{w_1}x_1(t)$$

Первый шаг в процессе корректировки модели – ее **параметрическая подстройка** без внесения в нее структурных изменений.

Полуэмпирические модели ДС (XLVI)

Экспериментальная оценка полуэмпирической модели – 12

Вычислительный эксперимент показывает, что **подстройка значения коэффициента w_1** во втором уравнении позволяет несколько улучшить точность модели, но **совершенно недостаточно**: ошибка моделирования снизилась с **0.1347** до **0.1278** для метода Эйлера и с **0.0434** до **0.0390** для метода Адамса при целевом значении **0.01**.

Поэтому **необходимо использовать второе предложение** из списка, приведенного выше: **заменить уравнение с линейной зависимостью** в его правой части от величины **x_1** на **уравнение с нелинейной зависимостью**.

Эта зависимость вводится в нейросетевую модель путем замены в ней фрагмента для упомянутой линейной зависимости (нейрон, отвечающий правой части уравнения), на нелинейную зависимость, которая задается **однослойной сетью с сигмоидальной активационной функцией** и 10 нейронами в слое (число нейронов подобрано в вычислительном эксперименте).

Полуэмпирические модели ДС (XLVII)

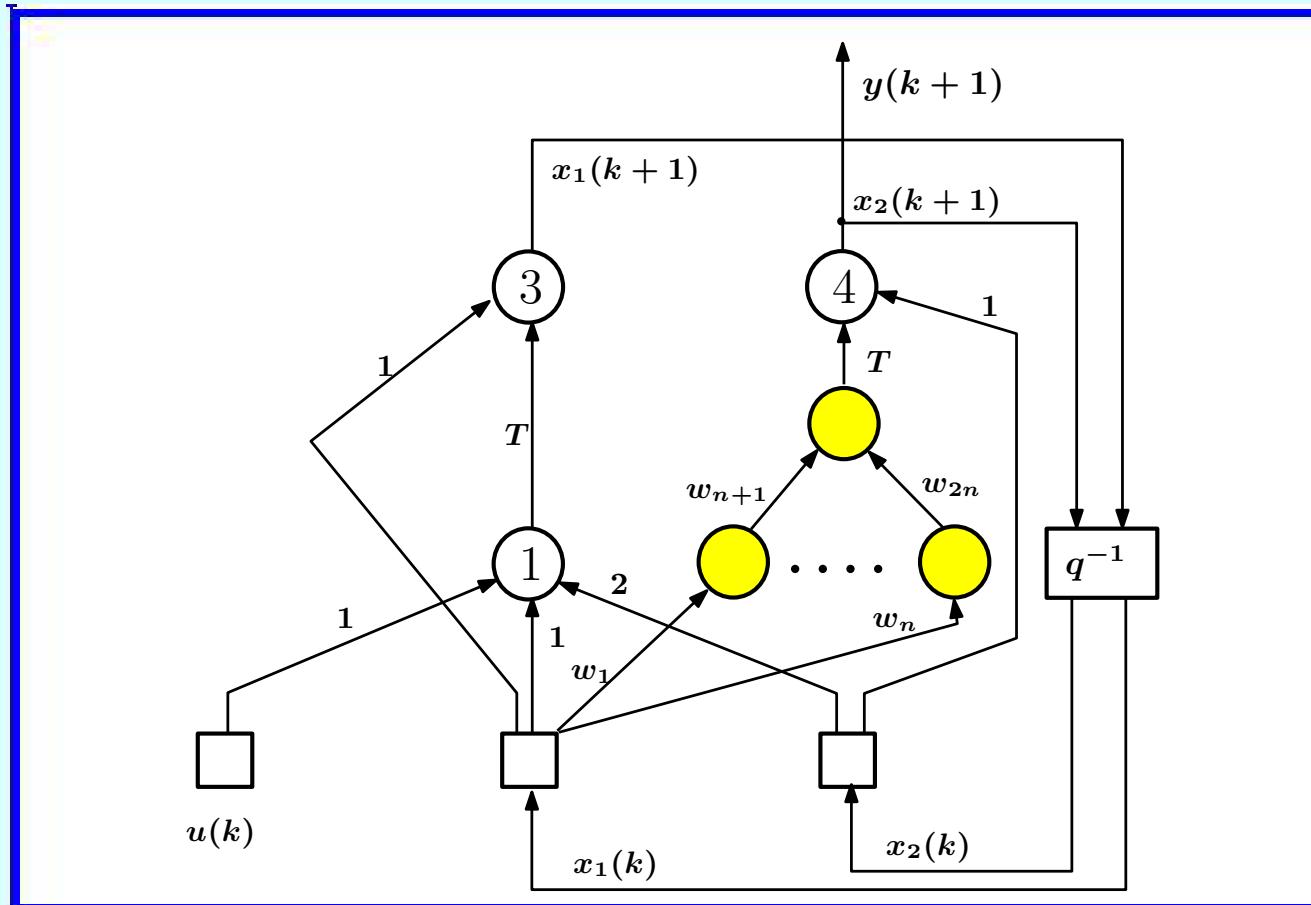
Экспериментальная оценка полуэмпирической модели – 13

Модельная задача **после первой структурной корректировки:**

$$\begin{aligned}\dot{x}_1(t) &= -(x_1(t) + 2x_2(t))^2 + u(t) \\ \dot{x}_2(t) &= \varphi(x_1(t))\end{aligned}$$

Полуэмпирические модели ДС (XLVIII)

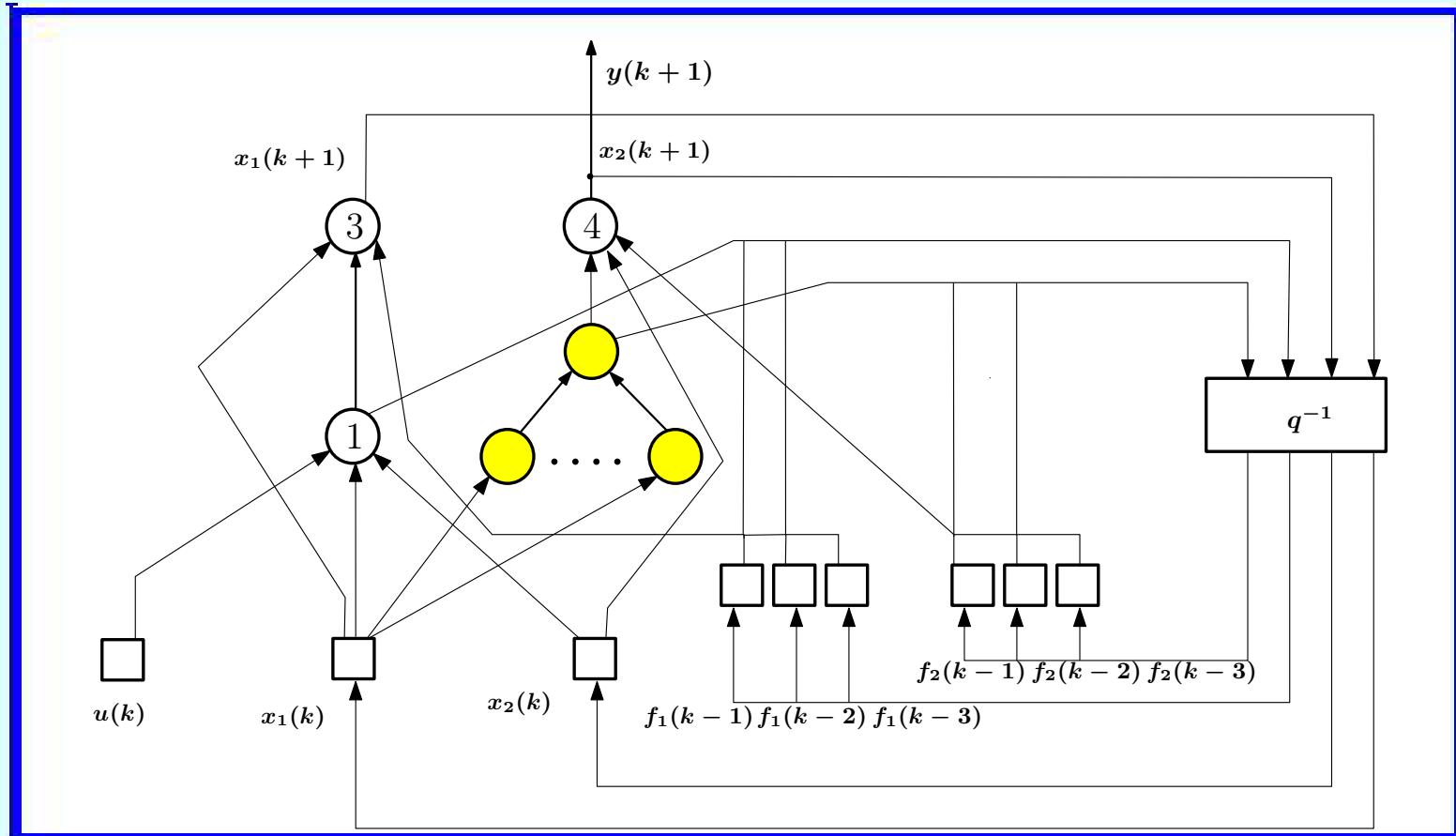
Экспериментальная оценка полуэмпирической модели – 14



Каноническая форма полуэмпирической модели (**метод Эйлера**), уточненная путем введения нелинейности по x_1 во второе уравнение

Полуэмпирические модели ДС (XLIX)

Экспериментальная оценка полуэмпирической модели – 15



Каноническая форма полуэмпирической модели (**метод Адамса**), уточненная
путем введения нелинейности по x_1 во второе уравнение

Полуэмпирические модели ДС (L)

Экспериментальная оценка полуэмпирической модели – 16

Введение нелинейности по x_1 во второе уравнение позволяет улучшить точность модели совсем незначительно, а именно, ошибка снизилась с **0.1278** до **0.1252** для метода Эйлера и с **0.0390** до **0.0375** для метода Адамса.

Это означает, что **только за счет подбора подходящей нелинейной зависимости от x_1** в правой части уравнения **требуемой точности добиться нельзя**, необходимо, очевидно, учитывать зависимость не только от x_1 , но и от x_2 , т. е. необходимо реализовать **третье предложение** из списка, представленного выше.

Эта зависимость вводится в нейросетевую модель **путем добавления в ее структуру связи** от x_2 к введенному на предыдущем шаге фрагменту модели в виде однослоиной сети.

Полуэмпирические модели ДС (LI)

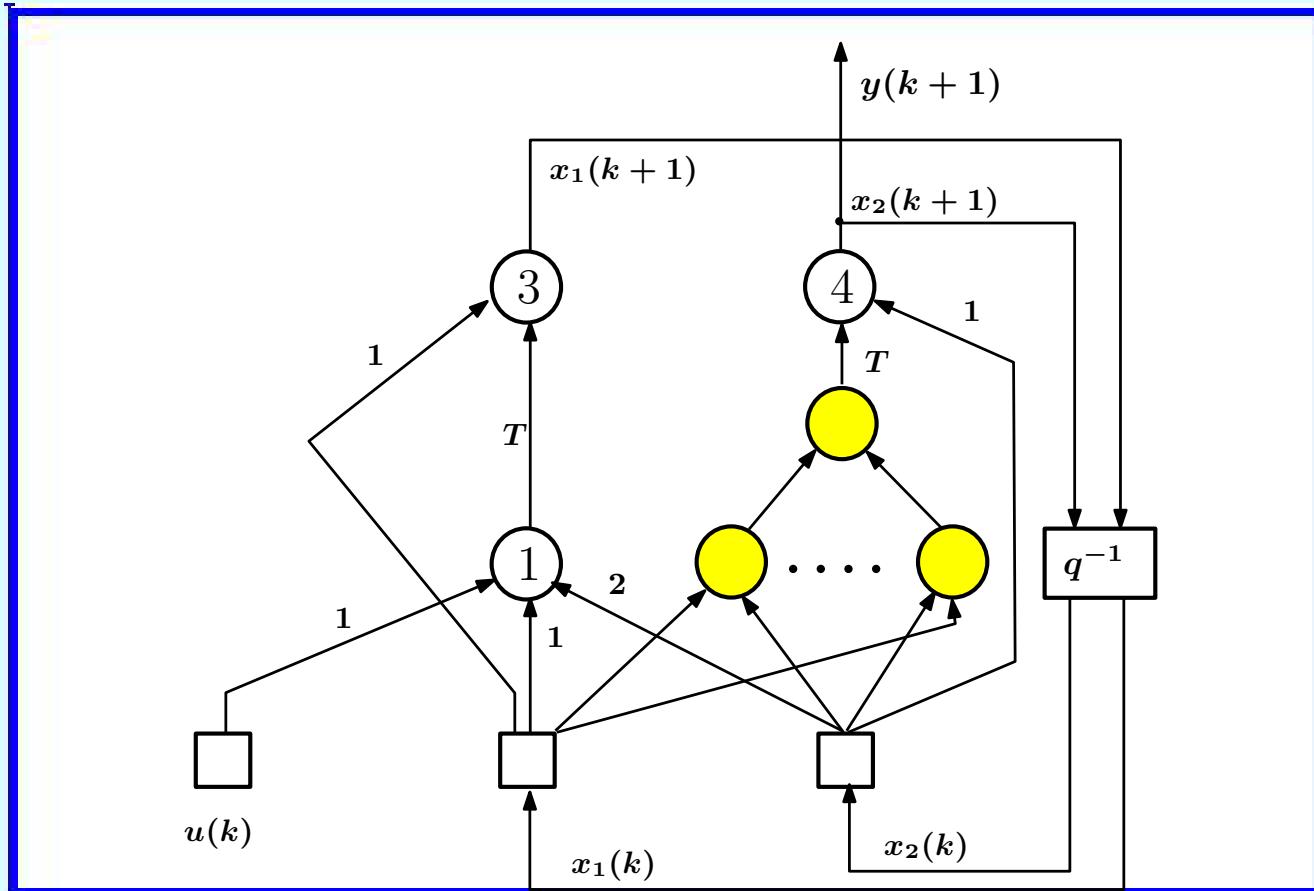
Экспериментальная оценка полуэмпирической модели – 17

Модельная задача **после второй структурной корректировки:**

$$\begin{aligned}\dot{x}_1(t) &= -(x_1(t) + 2x_2(t))^2 + u(t) \\ \dot{x}_2(t) &= \psi(x_1(t), x_2(t))\end{aligned}$$

Полуэмпирические модели ДС (ЛII)

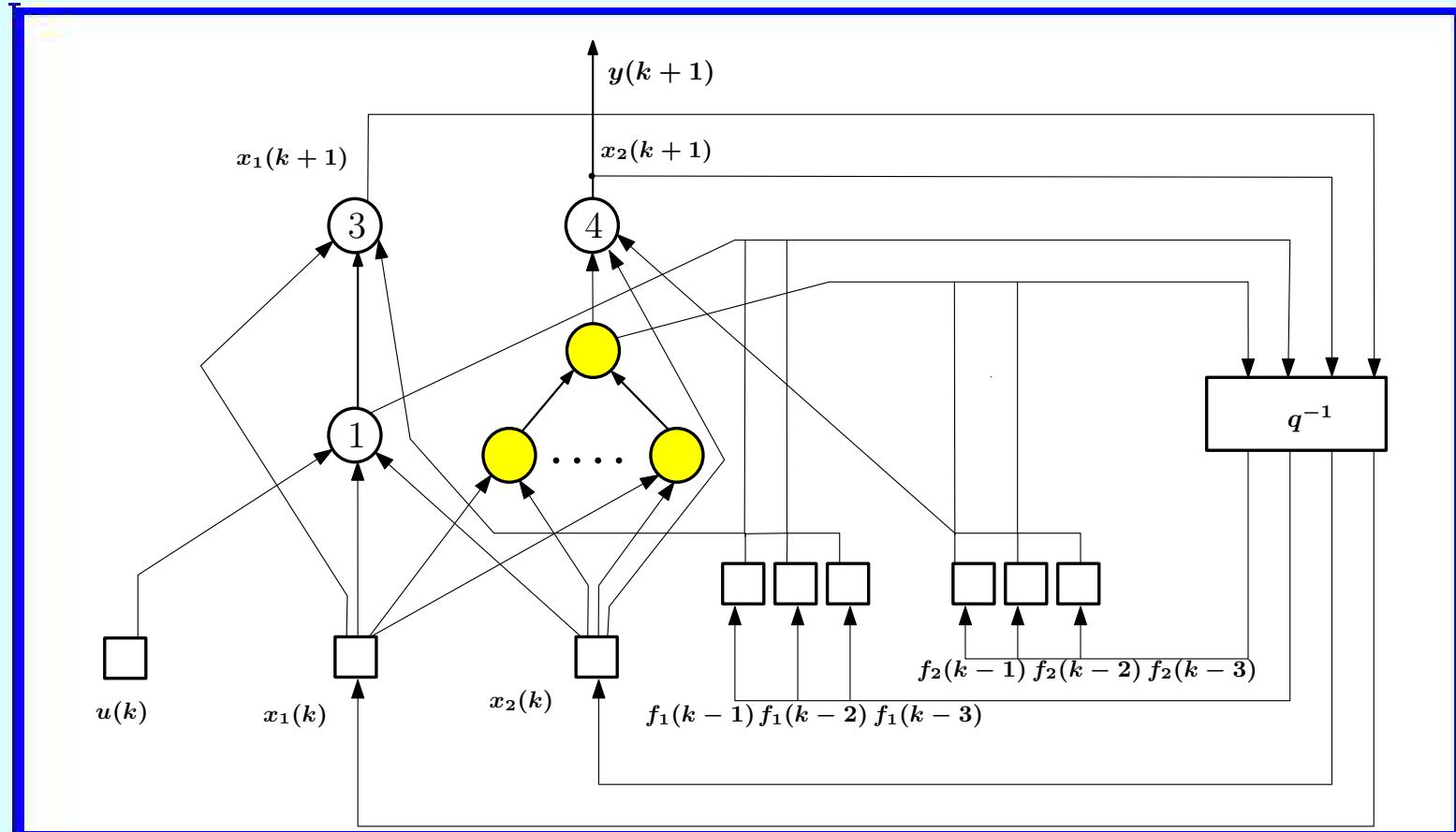
Экспериментальная оценка полуэмпирической модели – 18



Каноническая форма полуэмпирической модели (**метод Эйлера**), уточненная путем введения нелинейности **по x_1 и по x_2** во второе уравнение

Полуэмпирические модели ДС (ЛIII)

Экспериментальная оценка полуэмпирической модели – 19



Каноническая форма полуэмпирической модели (метод Адамса), уточненная
путем введения нелинейности по x_1 и по x_2 во второе уравнение

Полуэмпирические модели ДС (LIV)

Экспериментальная оценка полуэмпирической модели – 20

Введение дополнительной связи по x_2 помимо связи по x_1 во второе уравнение позволило решить поставленную задачу.

Точность модели теперь составляет **0.0141** для метода Эйлера и **0.0119** для метода Адамса.

Полуэмпирические модели ДС (LV)

Экспериментальная оценка полуэмпирической модели – 21

Для сравнения с традиционным подходом, продемонстрируем результаты обучения **NARX**-модели.

Лучшая точность **0.0258** в данном примере была достигнута сетью с 3 нейронами и 5 задержками в обратной связи.

Из этого сравнения видно **неоспоримое преимущество полуэмпирической модели над эмпирической** по достижимой точности решения: **даже для метода Эйлера** точность составляет **0.0141** против **0.0258** для NARX, **в случае метода Адамса** точность еще выше — она равна **0.0119**, при этом возможно еще большее повышение точности за счет применения более эффективных разностных схем.

Полуэмпирические модели ДС (LVI)

Экспериментальная оценка полуэмпирической модели – 22

Сопоставление результатов моделирования (задача 1)

	ОДУ	НС-1	НС-2	НС-3	Opt
Эйлер	0.13947	0.13593	0.12604	0.01394	–
Адамс	0.07143	0.07104	0.03883	0.01219	–
NARX	–	–	–	–	0.02821

Здесь: **ОДУ** — результаты, полученные непосредственным интегрированием исходной системы дифференциальных уравнений; **НС-1, НС-2, НС-3** — результаты для НС-моделей, полученных после первого, второго и третьего шагов корректировки, соответственно), а также с помощью модели типа **NARX** (четвертый столбец).

Полуэмпирические модели ДС (LVII)

Экспериментальная оценка полуэмпирической модели – 23

Модельный пример динамической системы (ДС)

(первый усложненный вариант):

$$\dot{x}_1(t) = -(x_1(t) + 2x_2(t))^2 + u(t)$$

$$\dot{x}_2(t) = 8.322109 \sin(x_1(t)) + 1.2 \cos(1.33\pi x_2(t))$$

Полуэмпирические модели ДС (LVIII)

Экспериментальная оценка полуэмпирической модели – 24

Сопоставление результатов моделирования (задача 2)

	ОДУ	НС-1	НС-2	НС-3	Opt
Эйлер	0.15684	0.15224	0.14079	0.01400	–
Адамс	0.07931	0.07858	0.05312	0.01185	–
NARX	–	–	–	–	0.03418

Здесь: **ОДУ** — результаты, полученные непосредственным интегрированием исходной системы дифференциальных уравнений; **НС-1, НС-2, НС-3** — результаты для НС-моделей, полученных после первого, второго и третьего шагов корректировки, соответственно), а также с помощью модели типа **NARX** (четвертый столбец).

Полуэмпирические модели ДС (LVIII)

Экспериментальная оценка полуэмпирической модели – 25

Модельный пример динамической системы (ДС)

(второй усложненный вариант):

$$\dot{x}_1(t) = -(x_1(t) + 2x_2(t))^2 + u(t)$$

$$\dot{x}_2(t) = 8.322109 \sin(x_1(t)) + 2 \cos(1.1\pi x_2(t))^2$$

Полуэмпирические модели ДС (LIX)

Экспериментальная оценка полуэмпирической модели – 26

Сопоставление результатов моделирования (задача 3)

	ОДУ	НС-1	НС-2	НС-3	Opt
Эйлер	0.18052	0.16880	0.15379	0.01272	–
Адамс	0.15394	0.14970	0.13111	0.01266	–
NARX	–	–	–	–	0.08403

Здесь: **ОДУ** — результаты, полученные непосредственным интегрированием исходной системы дифференциальных уравнений; **НС-1, НС-2, НС-3** — результаты для НС-моделей, полученных после первого, второго и третьего шагов корректировки, соответственно), а также с помощью модели типа **NARX** (четвертый столбец).

Полуэмпирические модели ДС (LX)

Экспериментальная оценка полуэмпирической модели – 27

	ОДУ	НС-1	НС-2	НС-3	Opt
Эйлер	0.13947	0.13593	0.12604	0.01394	–
Адамс	0.07143	0.07104	0.03883	0.01219	–
NARX	–	–	–	–	0.02821

	ОДУ	НС-1	НС-2	НС-3	Opt
Эйлер	0.15684	0.15224	0.14079	0.01400	–
Адамс	0.07931	0.07858	0.05312	0.01185	–
NARX	–	–	–	–	0.03418

	ОДУ	НС-1	НС-2	НС-3	Opt
Эйлер	0.18052	0.16880	0.15379	0.01272	–
Адамс	0.15394	0.14970	0.13111	0.01266	–
NARX	–	–	–	–	0.08403

Полуэмпирические модели ДС (LXI)

Полуэмпирические модели

Dreyfus G. Neural networks: Methodology and applications. – Berlin ao.: Springer, 2005. – xviii+497 pp.

Oussar Y., Dreyfus G. How to be a gray box: Dynamic semi-phisical modeling // *Neural Networks*. – 2001. – v. 14, No. 9. – pp. 1161–1172.

Dreyfus G., Idan Y. The canonical form of nonlinear discrete-time models // *Neural Computation*. – 1998. – v. 10. – pp. 133–164.

Обучающие наборы для динамических НС-моделей (I)

Проблема формирования обучающего набора для динамических НС-моделей – 1

Динамическая система — модель в пространстве состояний:

$$\begin{aligned}\dot{x} &= f(x, u, t), \\ y &= \varphi(x, u, t).\end{aligned}$$

Проблема обеспечения данными процесса обучения динамической НС-модели:

Моделируемая динамическая система часто характеризуется **сложным поведением**, а также существенной **многорежимностью**.

Сложное поведение:

- 1) очень большое **число фазовых траекторий** $x(t)$, потенциально реализуемых моделируемой системой;
- 2) большие **скорости изменения значений** $\dot{x}(t)$ фазовых переменных («энергичное маневрирование»).

Многорежимность:

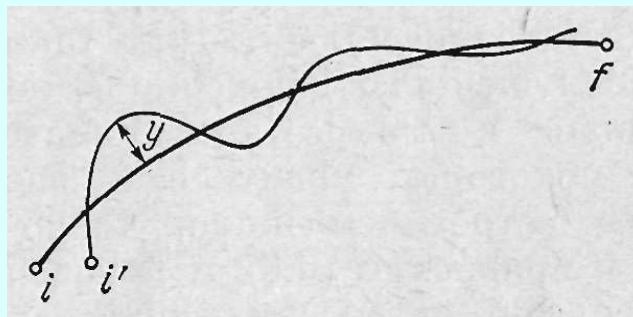
«Большие» **области изменения значений** фазовых переменных $x \in X$ и управляющих переменных $u \in U$.

Обучающие наборы для динамических НС-моделей (II)

Проблема формирования обучающего набора для динамических НС-моделей – 2

Формирование репрезентативного набора данных,

характеризующего поведение моделируемой ДС – одна из **критически важных** задач при синтезе как эмпирических, так и полуэмпирических НС-моделей.



Будем считать, что **действительное движение** объекта управления (x, u) складывается из **программного** (опорного) движения (x^*, u^*) и **возмущенного** движения (y, ξ):

$$x = x^* + y, \quad u = u^* + \xi.$$

Примеры опорного движения: прямолинейный горизонтальный полет самолета с постоянной скоростью, полет самолета с монотонно увеличивающимся углом атаки.

Средство получения возмущенного движения: тестовые входные управляющие (возмущающие) воздействия $\xi(t)$.

Результат возмущающего воздействия: данные $(u(t), x(t))$ для формирования обучающего набора $\langle u(t_i), x(t_i) \rangle, i = 0, 1, \dots, N$.

Обучающие наборы для динамических НС-моделей (III)

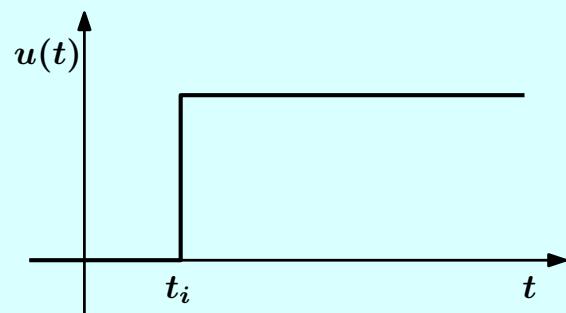
Виды тестовых сигналов для получения данных о поведении ДС – 1

В практике решения задач идентификации для управляемых динамических систем используются **типовые тестовые возмущающие воздействия**. Среди них наиболее употребительными являются следующие воздействия:

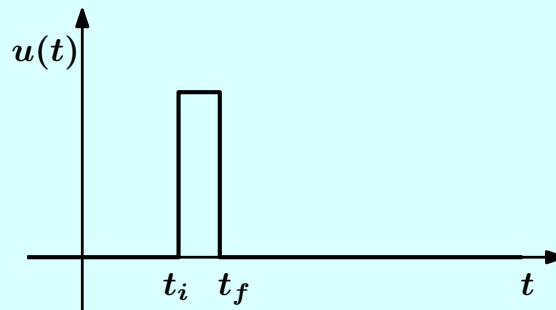
- (a) ступенчатое воздействие;
- (b) импульсное воздействие;
- (c) дублет (сигнал типа 1–1);
- (d) триплет (сигнал типа 2–1–1);
- (e) расширенный триплет (сигнал типа 2–2–1–1–1–1);
- (f) квадруплет (сигнал типа 3–2–1–1);
- (g) расширенный квадруплет (сигнал типа 3–3–2–2–1–1–1–1);
- (h) случайный сигнал;
- (i) полигармонический сигнал.

Обучающие наборы для динамических НС-моделей (IV)

Виды тестовых сигналов для получения данных о поведении ДС – 2



(a)

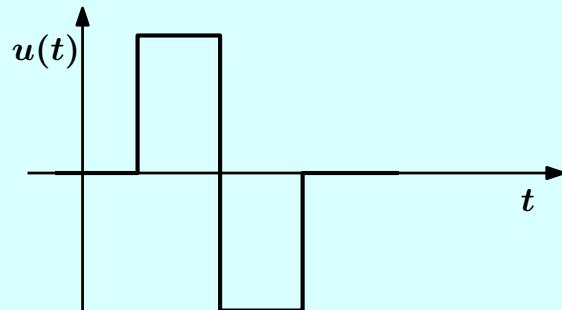


(b)

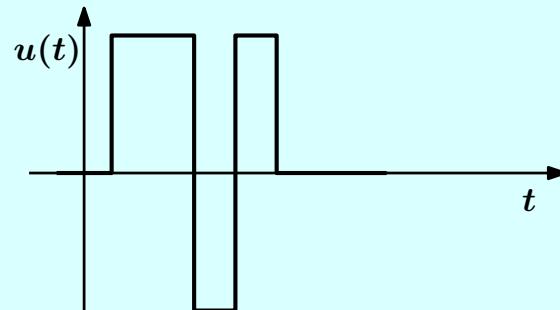
Типовые тестовые возмущающие воздействия, используемые при изучении динамики управляемых систем: (a) — **ступенчатое** воздействие;
(b) — **импульсное** воздействие

Обучающие наборы для динамических НС-моделей (V)

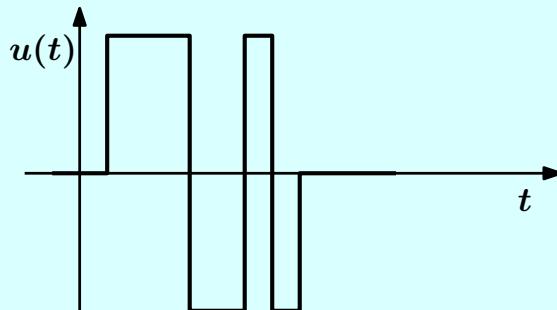
Виды тестовых сигналов для получения данных о поведении ДС – 3



(c)



(d)

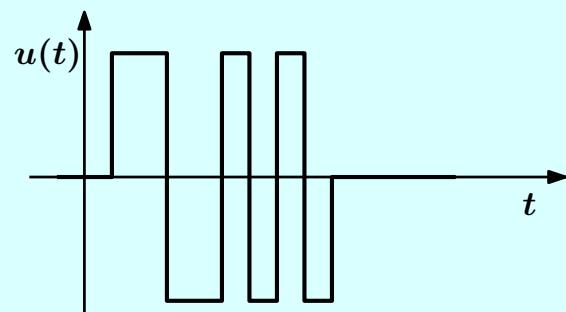


(e)

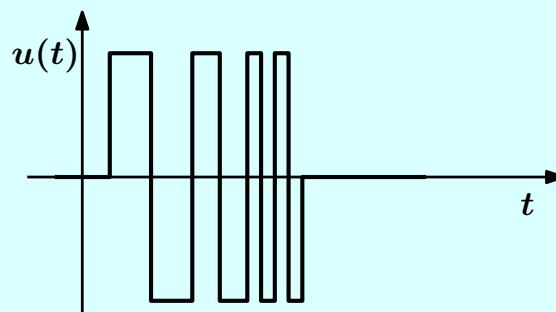
Типовые тестовые возмущающие воздействия, используемые при изучении динамики управляемых систем: (с) – **дублет** (сигнал типа 1–1); (д) – **триплет** (сигнал типа 2–1–1); (е) – **квадруплет** (сигнал типа 3–2–1–1)

Обучающие наборы для динамических НС-моделей (VI)

Виды тестовых сигналов для получения данных о поведении ДС – 4



(f)



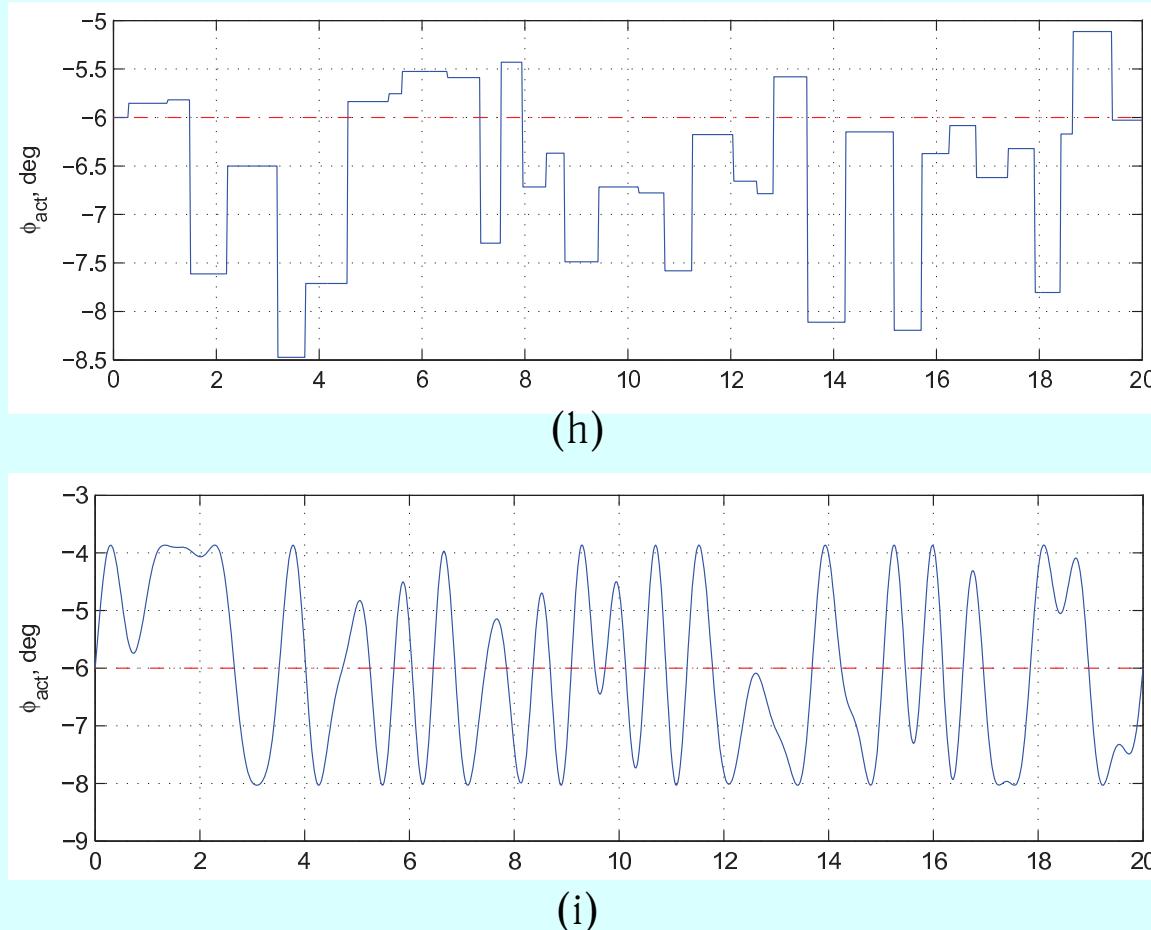
(g)

Типовые тестовые возмущающие воздействия, используемые при изучении
динамики управляемых систем:

- (f) — **расширенный триплет** (сигнал типа 2–2–1–1–1–1);
- (g) — **расширенный квадруплет** (сигнал типа 3–3–2–2–1–1–1–1)

Обучающие наборы для динамических НС-моделей (VII)

Виды тестовых сигналов для получения данных о поведении ДС – 5



Тестовые возмущающие воздействия, используемые при изучении динамики управляемых систем: (h) — **случайный** сигнал; (i) — **полигармонический** сигнал. Здесь φ_{act} — командный сигнал привода руля высоты (цельноповоротного горизонтального оперения)

Обучающие наборы для динамических НС-моделей (VIII)

Полигармонический тестовый сигнал для ДС – 1

Входное воздействие (multisine excitation) для каждого из m органов управления ДС формируется как **сумма гармонических сигналов**, каждый из которых обладает своим собственным сдвигом по фазе φ_k . Входное воздействие u_j , отвечающее j -му органу управления, имеет вид:

$$u_j = \sum_{k \in I_k} A_k \cos\left(\frac{2\pi k t}{T} + \varphi_k\right), \quad j = 1, \dots, m, \quad I_k \subset K, \quad K = \{1, 2, \dots, M\}$$

Здесь M — общее число гармонически связанных частот; T — промежуток времени, в течение которого на ДС действует тестовый возбуждающий сигнал; A_k — амплитуда k -й синусоидальной компоненты; время t принимается дискретным, т.е.

$$t = \{t(0), t(1), \dots, t(i), \dots, t(N - 1)\},$$

$$u_j = \{u_j(0), u_j(1), \dots, u_j(i), \dots, u_j(N - 1)\}, \quad u_j(i) = u_j(t(i)).$$

Обучающие наборы для динамических НС-моделей (IX)

Полигармонический тестовый сигнал для ДС – 2

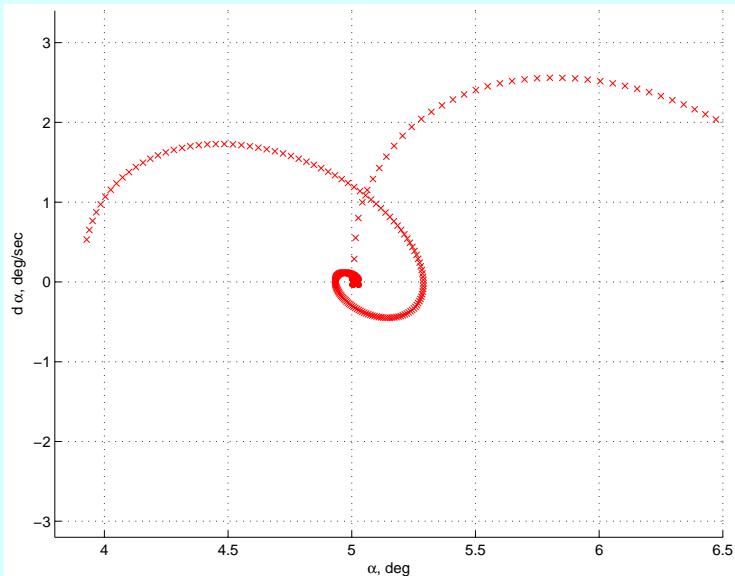
Тестовая модель ДС: модель углового продольного движения в виде, традиционном для динамики полета самолетов:

$$\begin{aligned}\dot{\alpha} &= \omega_z - \frac{qS}{mV} C_{y_a}(\alpha, \omega_z, \varphi) + \frac{g}{V}, \\ \dot{\omega}_z &= \frac{qSb_A}{J_{zz}} m_z(\alpha, \omega_z, \varphi), \\ T^2 \ddot{\varphi} &= -2T\zeta\dot{\varphi} - \varphi + \varphi_{act}.\end{aligned}$$

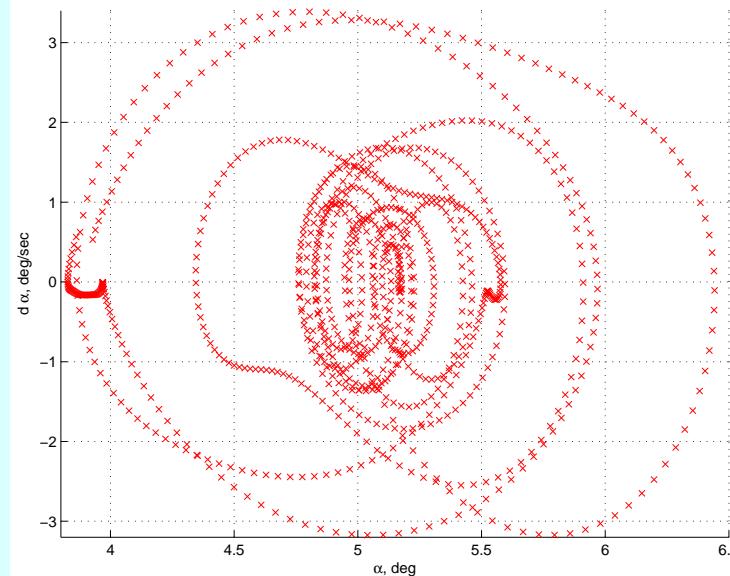
Здесь α — угол атаки, град; ω_z — угловая скорость тангажа, град/с; φ — угол отклонения управляемого стабилизатора, град; C_{y_a} — коэффициент подъемной силы; m_z — коэффициент момента тангажа; m — масса самолета, кг; V — воздушная скорость, м/с; $q = \rho V^2 / 2$ — скоростной напор; ρ — плотность воздуха, кг/м³; g — ускорение силы тяжести, м/с²; S — площадь крыла, м²; b_A — средняя аэродинамическая хорда крыла, м; J_{zz} — момент инерции самолета относительно боковой оси, кг·м²; безразмерные коэффициенты C_{y_a} и m_z являются нелинейными функциями своих аргументов; T , ζ — постоянная времени и коэффициент относительного демпфирования привода, φ_{act} — командный сигнал на привод цельноповоротного управляемого стабилизатора.

Обучающие наборы для динамических НС-моделей (Х)

Репрезентативность обучающего набора – 1



(a)

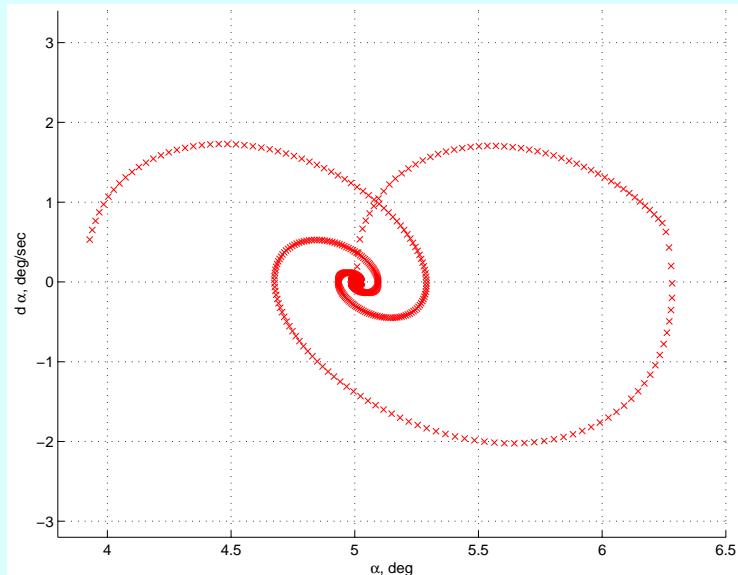


(b)

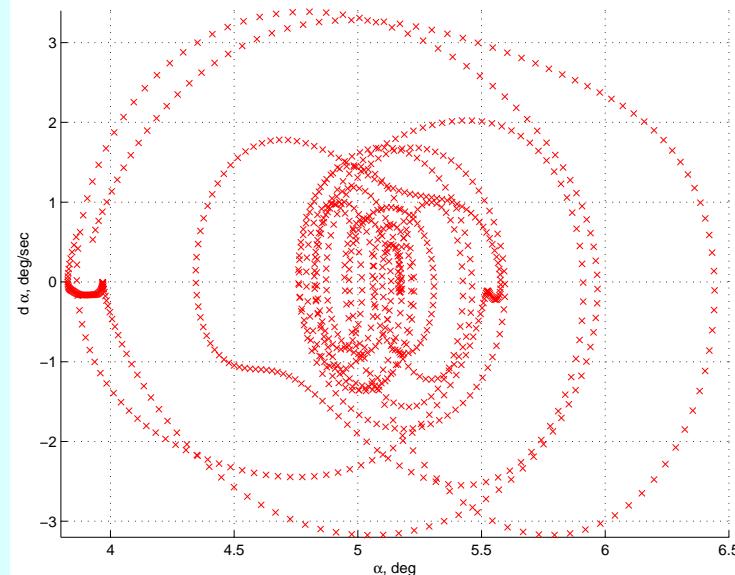
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «ступенчатый» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XI)

Репрезентативность обучающего набора – 2



(a)

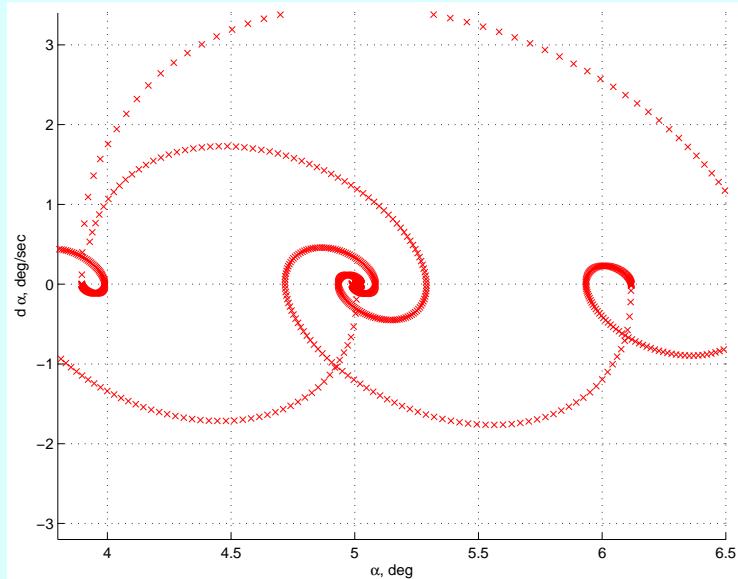


(b)

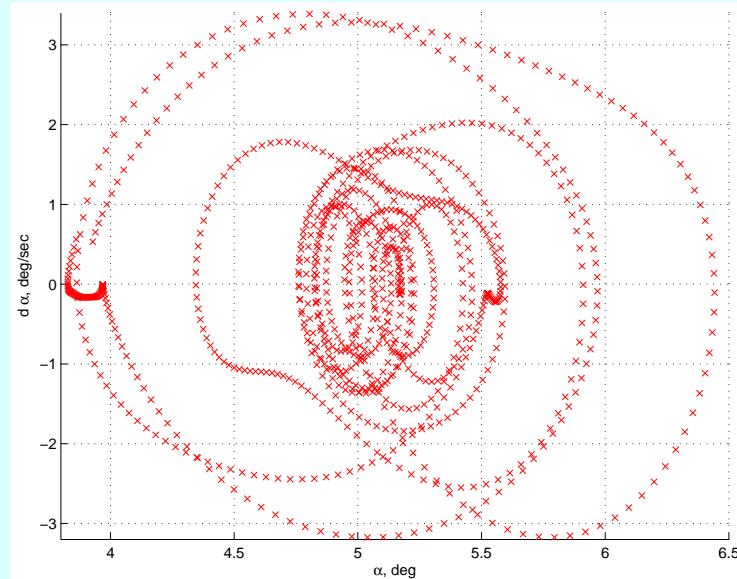
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «импульс» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XII)

Репрезентативность обучающего набора – 3



(a)

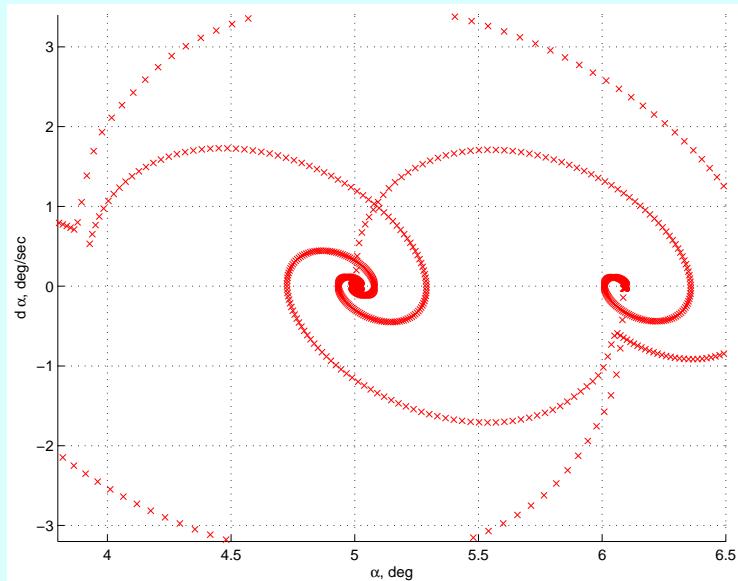


(b)

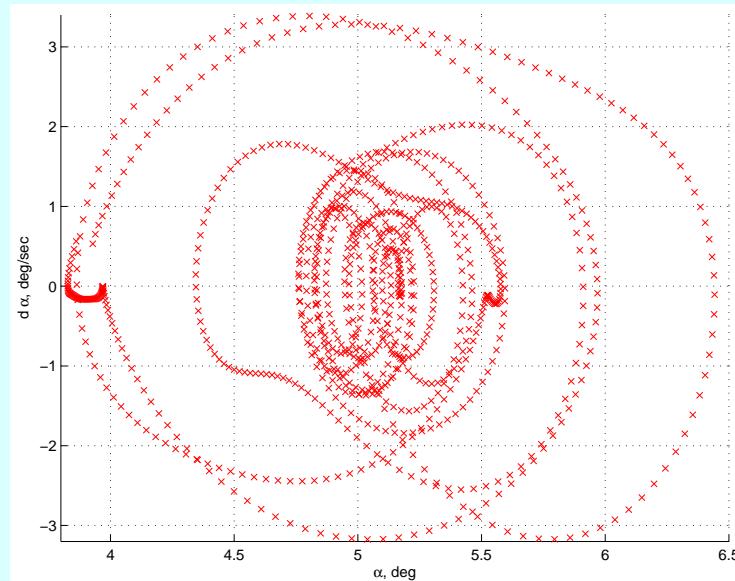
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «дублет» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XIII)

Репрезентативность обучающего набора – 4



(a)

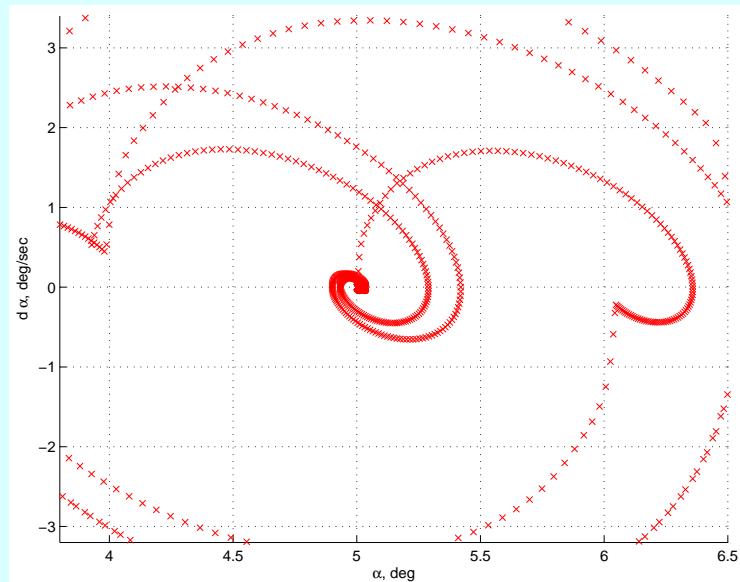


(b)

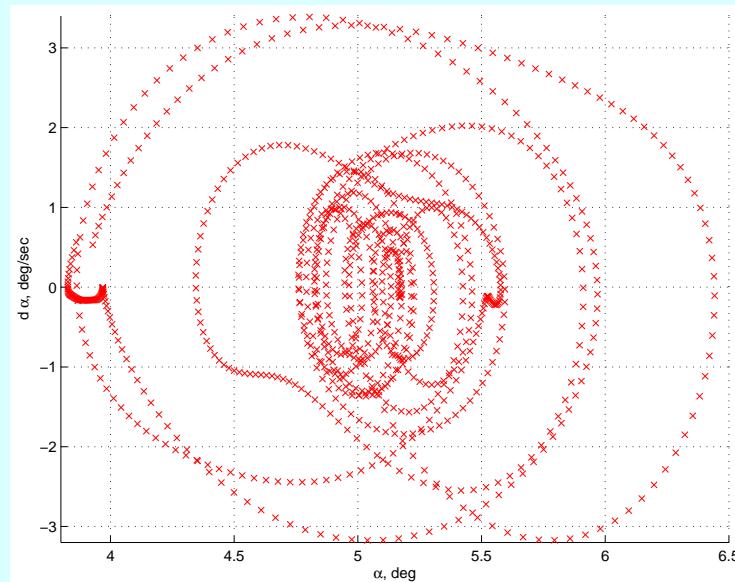
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «триплет» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XIV)

Репрезентативность обучающего набора – 5



(a)

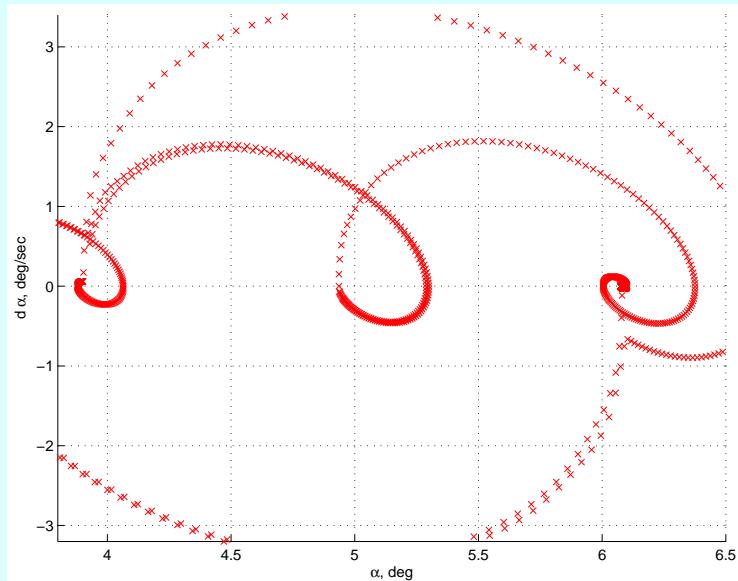


(b)

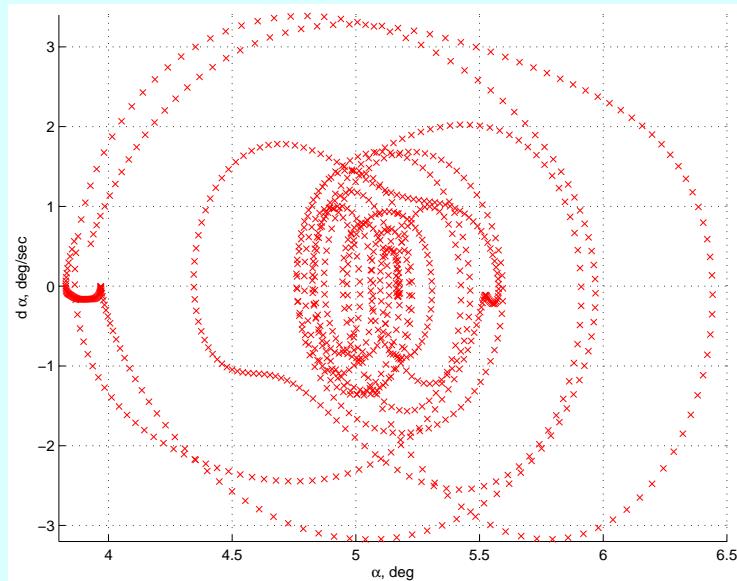
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «расширенный триплет» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XV)

Репрезентативность обучающего набора – 6



(a)

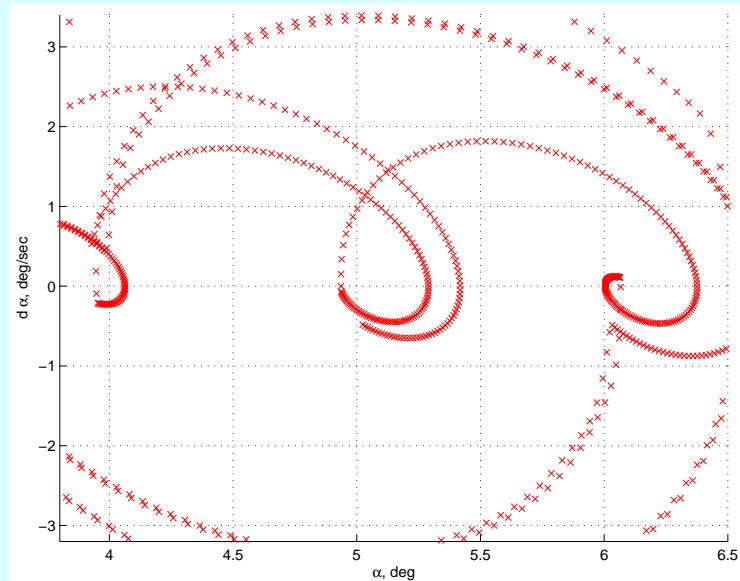


(b)

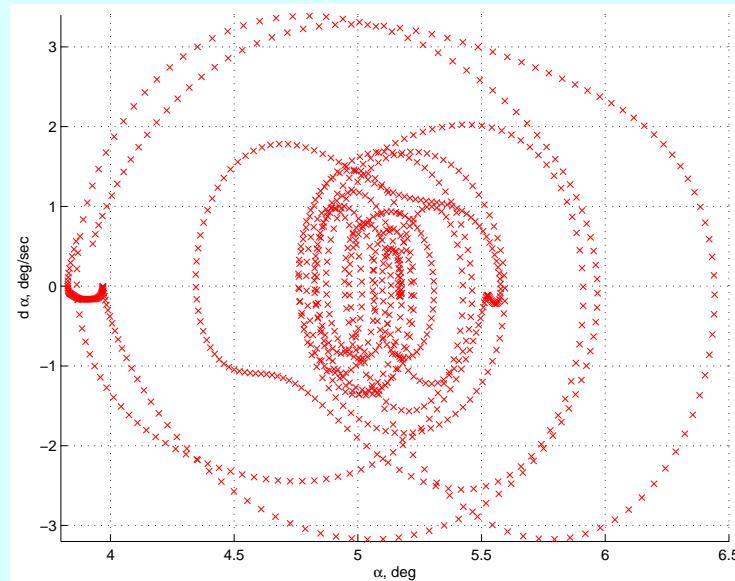
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «квадруплет» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XVI)

Репрезентативность обучающего набора – 7



(a)

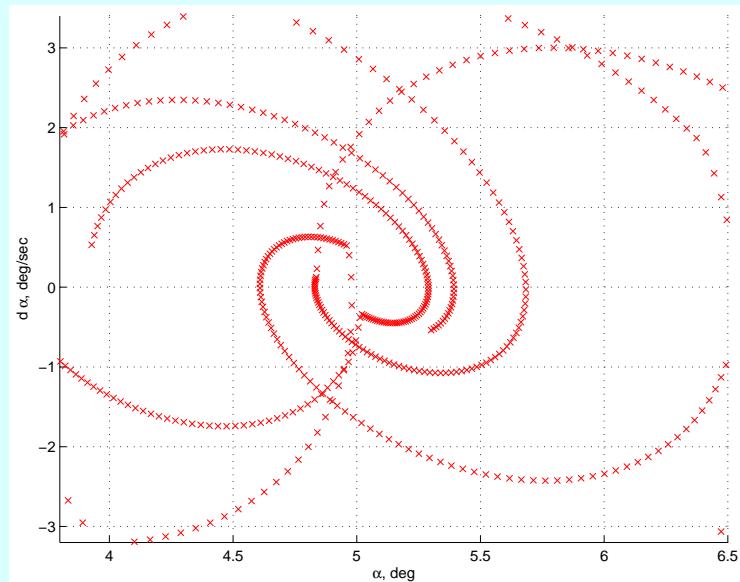


(b)

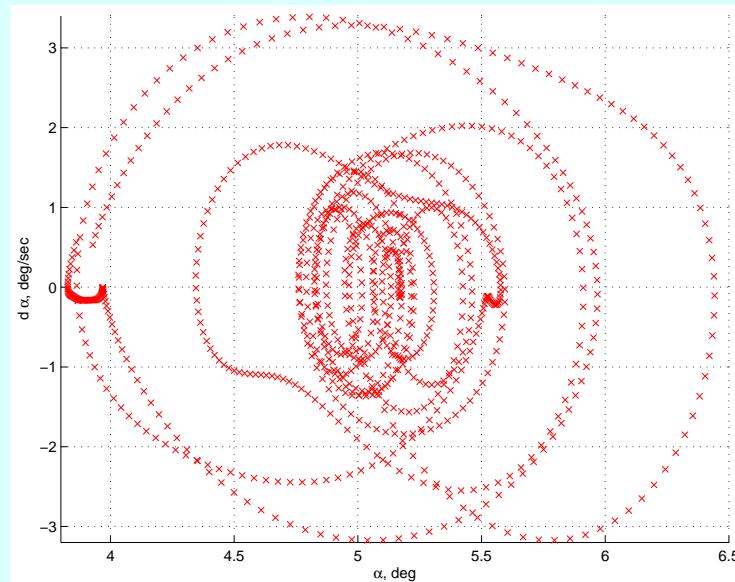
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «расширенный квадруплет» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XVII)

Репрезентативность обучающего набора – 8



(a)

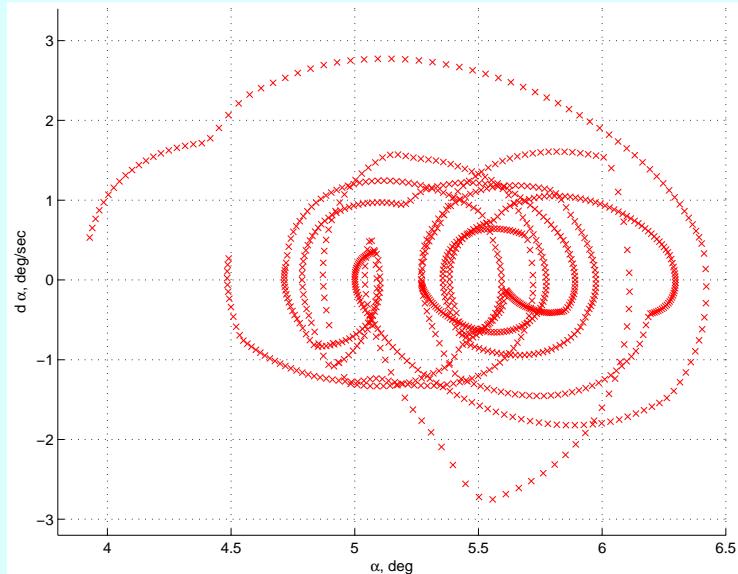


(b)

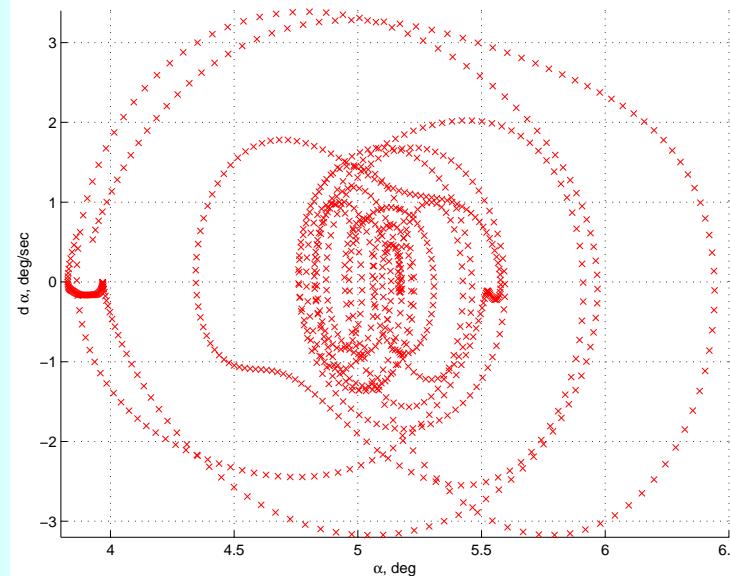
Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «триплет+квадруплет» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

Обучающие наборы для динамических НС-моделей (XVIII)

Репрезентативность обучающего набора – 9



(a)

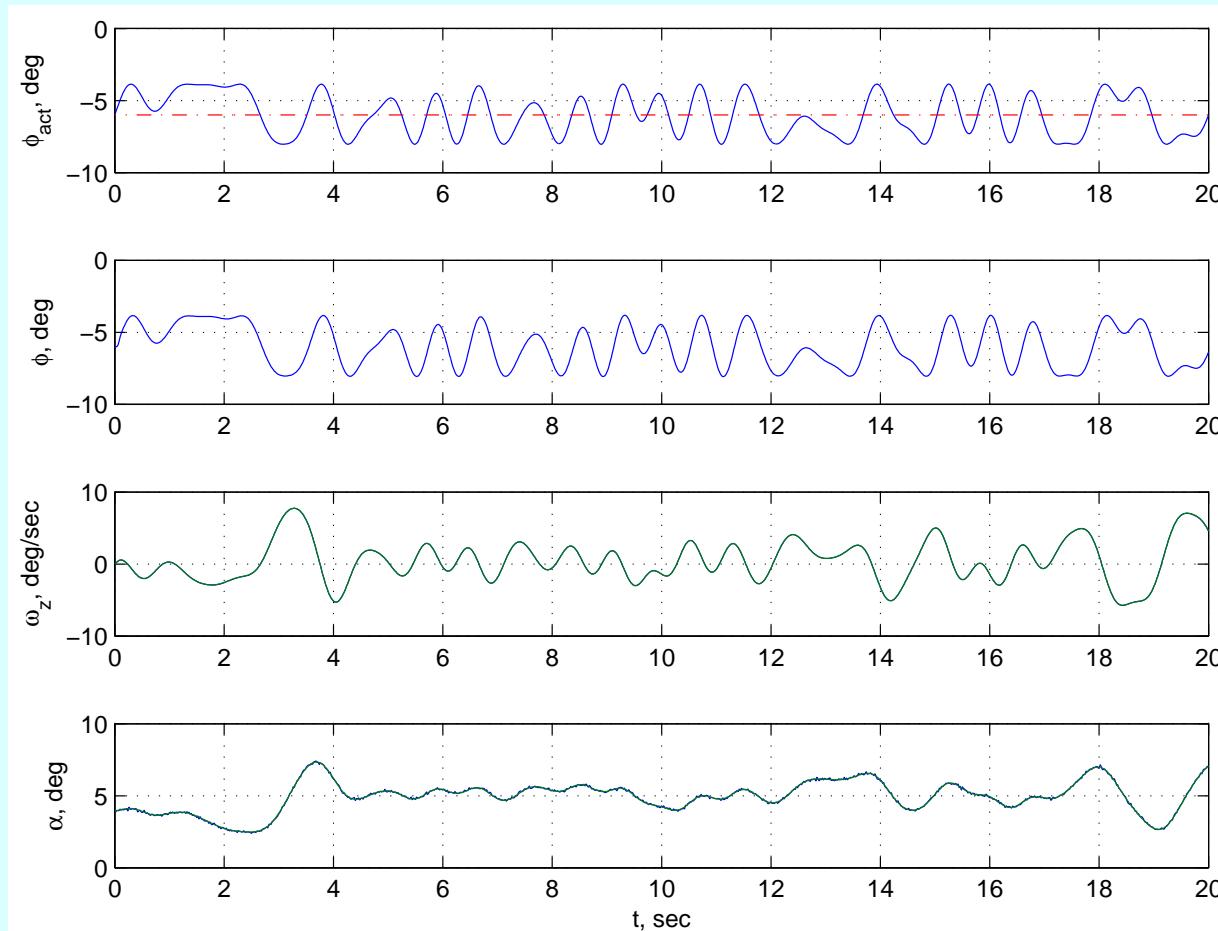


(b)

Диаграммы представительности $(\alpha, \dot{\alpha})$ обучающего набора при воздействиях типа «случайный» (а) и «полигармонический» (б) при равном числе обучающих примеров ($N = 1000$)

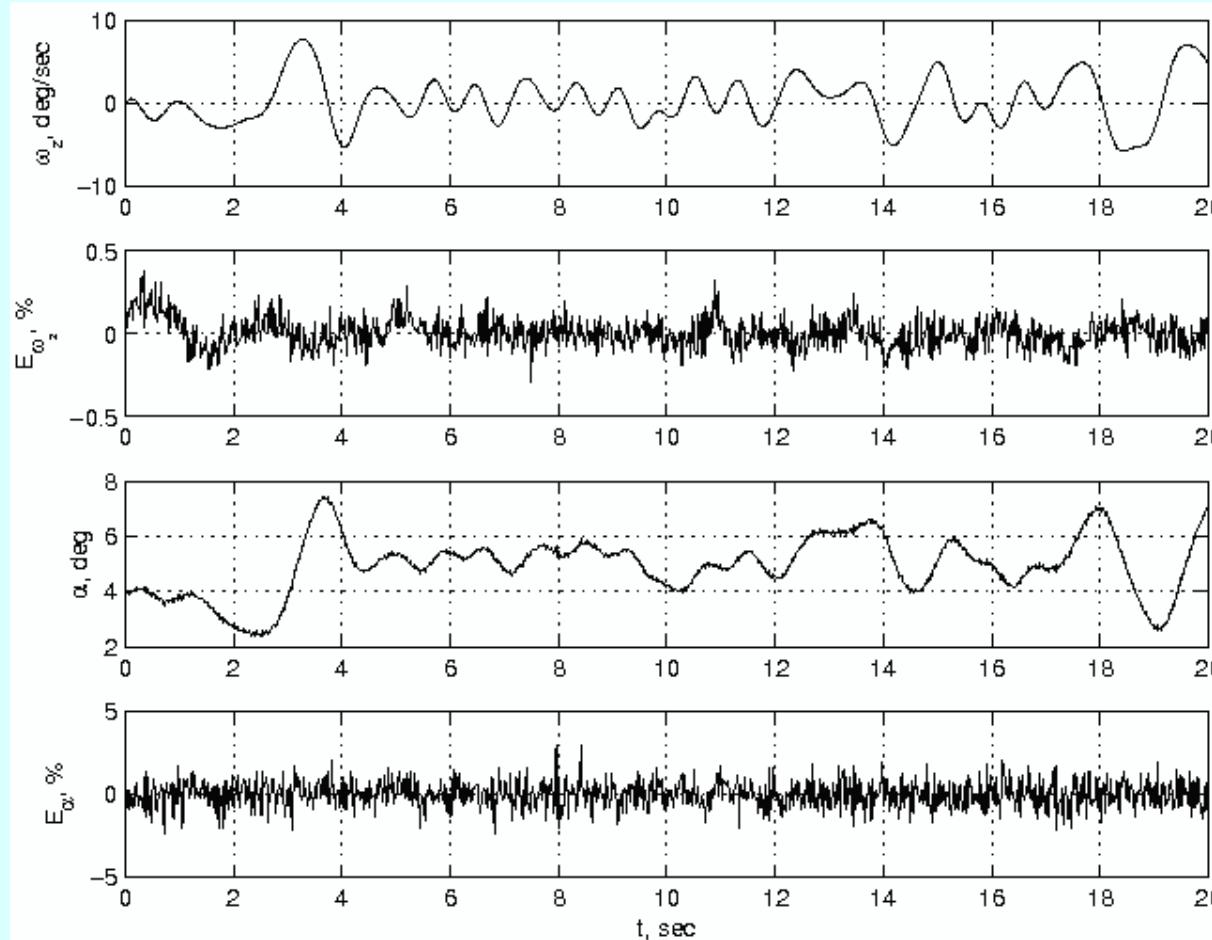
Обучающие наборы для динамических НС-моделей (XIX)

Результативность использования полигармонического сигнала – 1



Точность восстановления зависимостей $C_{y_\alpha}(\alpha)$ и $m_z(\alpha)$ по результатам тестирования
НС-модели (точечный режим, идентификация и тестирование — полигармонический сигнал)

Обучающие наборы для динамических НС-моделей (ХХ) Результативность использования полигармонического сигнала – 2



Точность восстановления зависимостей $C_{y_\alpha}(\alpha)$ и $m_z(\alpha)$ по результатам тестирования
НС-модели (точечный режим, идентификация и тестирование — полигармонический сигнал)

Обучающие наборы для динамических НС-моделей (ХХI)

Результативность использования полигармонического сигнала – 3

Таблица 1

Ошибка моделирования на обучающем множестве (полигармонический сигнал)

	Точечный режим		Монотонный режим	
	СКО _α	СКО _{ωz}	СКО _α	СКО _{ωz}
Дообучение C_y	1.02e-3	1.23e-4	1.02e-3	1.24e-4
Обучение C_y	1.02e-3	1.23e-4	1.02e-3	1.24e-4
Обучение C_y, m_z	1.02e-3	1.19e-4	1.02e-3	1.27e-4
NARX	1.85e-3	3.12e-3	1.12e-3	7.36e-4

Таблица 2

Ошибка моделирования на тестовом множестве (полигармонический сигнал)

	Точечный режим		Монотонный режим	
	СКО _α	СКО _{ωz}	СКО _α	СКО _{ωz}
Дообучение C_y	1.02e-3	1.59e-4	1.02e-3	1.17e-4
Обучение C_y	1.02e-3	1.59e-4	1.02e-3	1.17e-4
Обучение C_y, m_z	1.02e-3	1.32e-4	1.02e-3	1.59e-4
NARX	2.32e-2	4.79e-2	3.16e-2	5.14e-2

Проблемы при обучения динамических НС-моделей (I)

Причины появления проблем

Рекуррентная нейронная сеть — **сложный для обучения объект**.

Основные источники трудностей:

- бифуркации режимов функционирования** («динамики») сети при изменении значений синаптических весов в процессе обучения НС-модели;
- наличие **долговременных зависимостей** (long-term dependencies) выходов сети от входов и состояний НС-модели в предыдущие моменты времени;
- очень сложный **рельеф функции ошибки**, изрезанный многочисленными глубокими, узкими и искривленными впадинами.

Проблемы при обучения динамических НС-моделей (II)

Бифуркации динамики сети – 1

Бифуркация — качественная перестройка режимов функционирования системы при малом изменении ее параметров.

Уравнение колебаний маятника:

$$\ddot{x} + \alpha \dot{x} + \omega_0^2 = 0$$

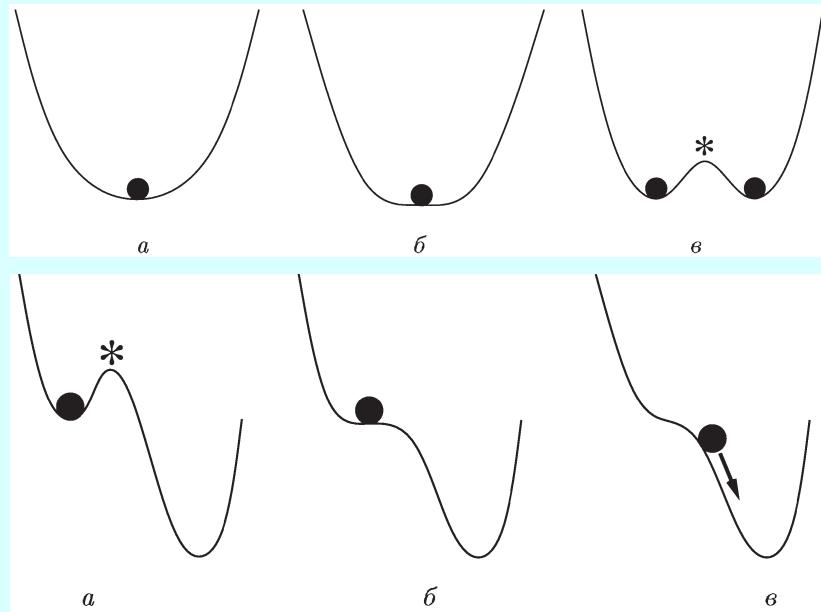
Параметры:

α — параметр затухания, характеризующий трение, ($0 < \alpha < 1$);

ω_0 — параметр, определяющий частоту колебаний

Проблемы при обучении динамических НС-моделей (III)

Бифуркации динамики сети – 2



Мягкая бифуркация (верхний рисунок): Стационарное состояние **(а)** теряет устойчивость **(б)** и вблизи него появляются два новых устойчивых стационарных состояния **(в)**; звездочкой помечено неустойчивое состояние.

Жесткая потеря устойчивости (нижний рисунок) стационарным состоянием: **(а)** при $\mu < \mu^*$ шарик находится в устойчивом стационарном состоянии, есть еще неустойчивое состояние (звездочка); **(б)** в точке бифуркации $\mu = \mu^*$ устойчивое и неустойчивое состояние сливаются в одно; **(в)** далее эти состояния исчезают и система выбирает новый режим, который существенно отличается от предыдущего и не находится в непосредственной близости от исходного режима.

Источник: Анищенко В. С. Знакомство с нелинейной динамикой. – Москва-Ижевск: Институт компьютерных исследований, 2002. – 144 с. (Рис.2.2 и 2.3, с.35, 36).

Бифуркация динамики сети – качественное изменение динамических свойств и характера поведения НС-модели при малых изменениях ее настраиваемых параметров (синаптических весов). В терминах нейросетевого обучения это означает, что **меняется рельеф функции ошибки**.

Проблемы при обучения динамических НС-моделей (IV)

Долговременные зависимости

При обучении динамических сетей существует **проблема долговременных зависимостей** (long-term dependencies), обусловленная тем, что выход НС-модели зависит от ее входов и состояний в предыдущие моменты времени, **далеко отстоящие от текущего момента времени**.

Градиентные методы поиска минимума функции ошибки ведут себя в этом случае **неудовлетворительно**.

Причина: Анализ асимптотического поведения ошибки обучения и ее градиента в процессе обратного распространения показывает, что их значения быстро (как правило, экспоненциально) убывают.

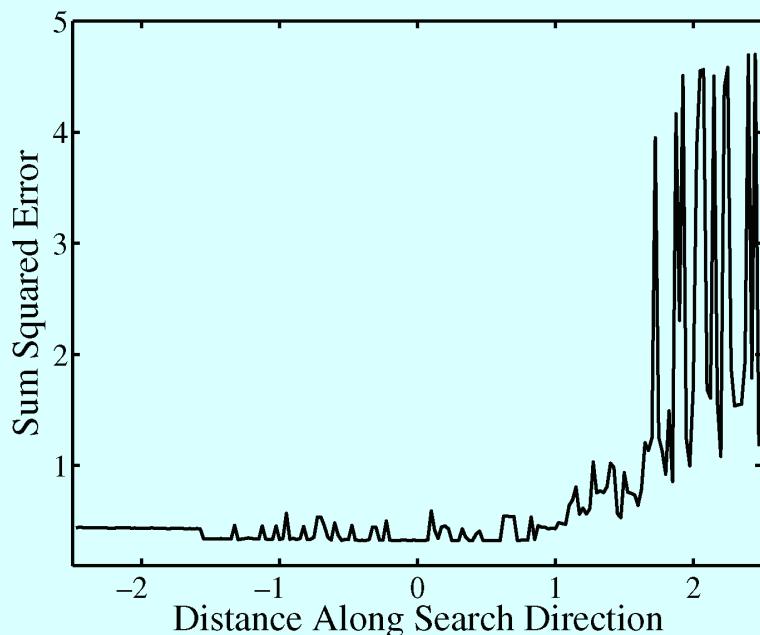
Обоснование см.: *Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J.* Gradient flow in recurrent nets: The difficulty of learning long-term dependencies // Chapter 14 in “A Field Guide to Dynamical Recurrent Networks” / Ed. by J.F.Kolen and S.C.Kremer. – IEEE Press: New York, 2001. – pp.237–243.

Schaefer A.M., Udluft S., Zimmermann H.-G. Learning long-term dependencies with recurrent neural networks // Neurocomputing. – 2008. – Vol.71, No.13–15. – pp.2481–2488.

Проблемы при обучении динамических НС-моделей (V)

Сложный рельеф функции ошибки – 1

Одна из важнейших причин возникновения трудностей при обучении динамических НС-моделей — очень сложный **рельеф функции ошибки**, изрезанный **многочисленными глубокими, узкими и искривленными впадинами**.



Пример: обучение нейроконтроллера

В рекуррентной НС-модели варьировались значения **65 весов**.

Квазиньютоновский **алгоритм обучения BFGS** (Broyden-Fletcher-Goldfarb-Shanno).

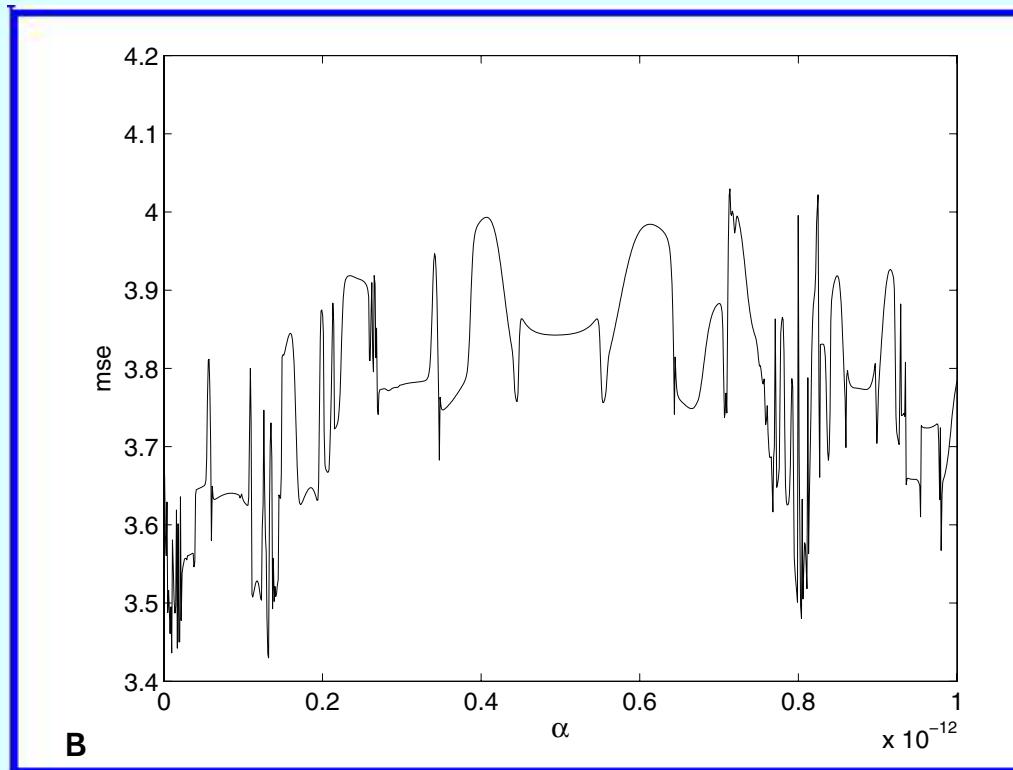
Ошибка обучения иногда **начинала расти**, несмотря на использование одномерной минимизации функции ошибки вдоль направления поиска.

Анализ показал **наличие узких долин** (с шириной порядка 10^{-10}), **заостряющихся книзу** несмотря на дифференцируемость функции ошибки.

Источник: Horn J., De Jesús O., Hagan M.T. Spurious valleys in the error surface of recurrent networks — analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Проблемы при обучении динамических НС-моделей (VI)

Сложный рельеф функции ошибки – 2



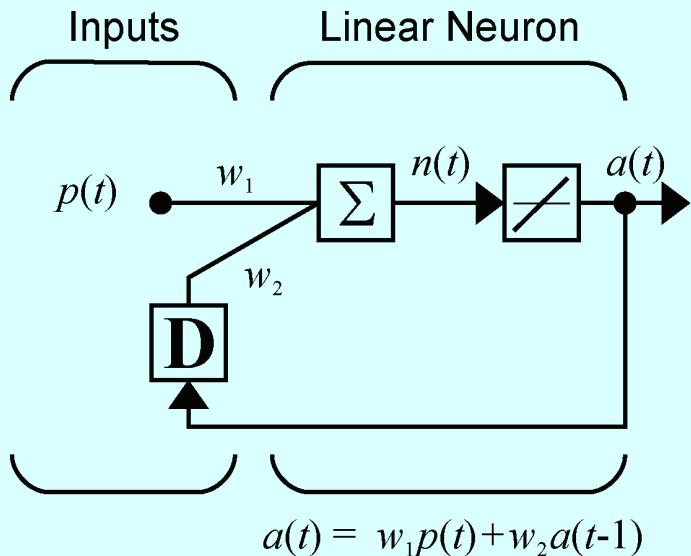
Профиль функции ошибки вдоль направления поиска

Источник: *Phan M.C., Hagan M.T.* Error surface of recurrent neural networks // IEEE Transactions on Neural Networks. – 2009. – Vol.24, No.11. – pp.1709–1721.

Проблемы при обучении динамических НС-моделей (VII)

Сложный рельеф функции ошибки – 3

Линейная рекуррентная сеть первого порядка (1)



Формирование обучающего набора: путем подачи на вход сети случайной последовательности $p(t)$ (гауссовский белый шум с нулевым средним и единичной дисперсией), порождающей последовательность выходов $a(t)$.

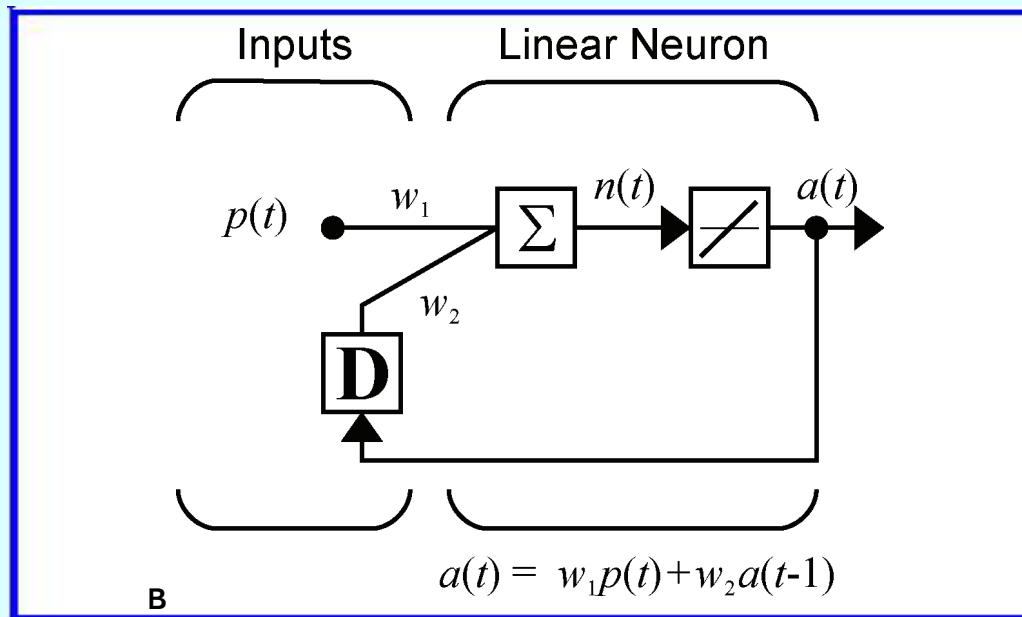
Значения весов: $w_1 = 0.5$, $w_2 = 0.5$.

Цель обучения: используя полученный обучающий набор $(p(t), a(t))$, обучить сеть для случайных стартовых значений весов w_1, w_2 .

Источник: Horn J., De Jesús O., Hagan M.T. Spurious valleys in the error surface of recurrent networks — analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Проблемы при обучении динамических НС-моделей (VIII) Сложный рельеф функции ошибки – 4

Линейная рекуррентная сеть первого порядка (2)



Один из источников, порождающих впадины на рельефе функции ошибки –
входные данные $p(k)$, $k = 0, 1, 2 \dots, n$.

Выход сети на шаге с номером k :

$$a(k) = w_1 p(k) + w_2 a(k - 1)$$

Проблемы при обучения динамических НС-моделей (IX)

Сложный рельеф функции ошибки – 5

Линейная рекуррентная сеть первого порядка (3)

Один из источников, порождающих впадины на рельефе функции ошибки – **входные данные** $p(k)$, $k = 0, 1, 2 \dots, n$.

Выход сети на шаге с номером k :

$$a(k) = w_1 p(k) + w_2 a(k - 1)$$

Накопление откликов сети, начиная со стартового состояния $k = 0$, до момента времени $k = n$:

$$k = 0 : a(0) = a(0)$$

$$k = 1 : a(1) = w_1 p(1) + w_2 a(0)$$

$$k = 2 : a(2) = w_1 p(2) + w_2 a(1) =$$

$$= w_1 p(2) + w_2 (w_1 p(1) + w_2 a(0)) =$$

$$= w_1 p(2) + w_2 w_1 p(1) + w_2^2 a(0)$$

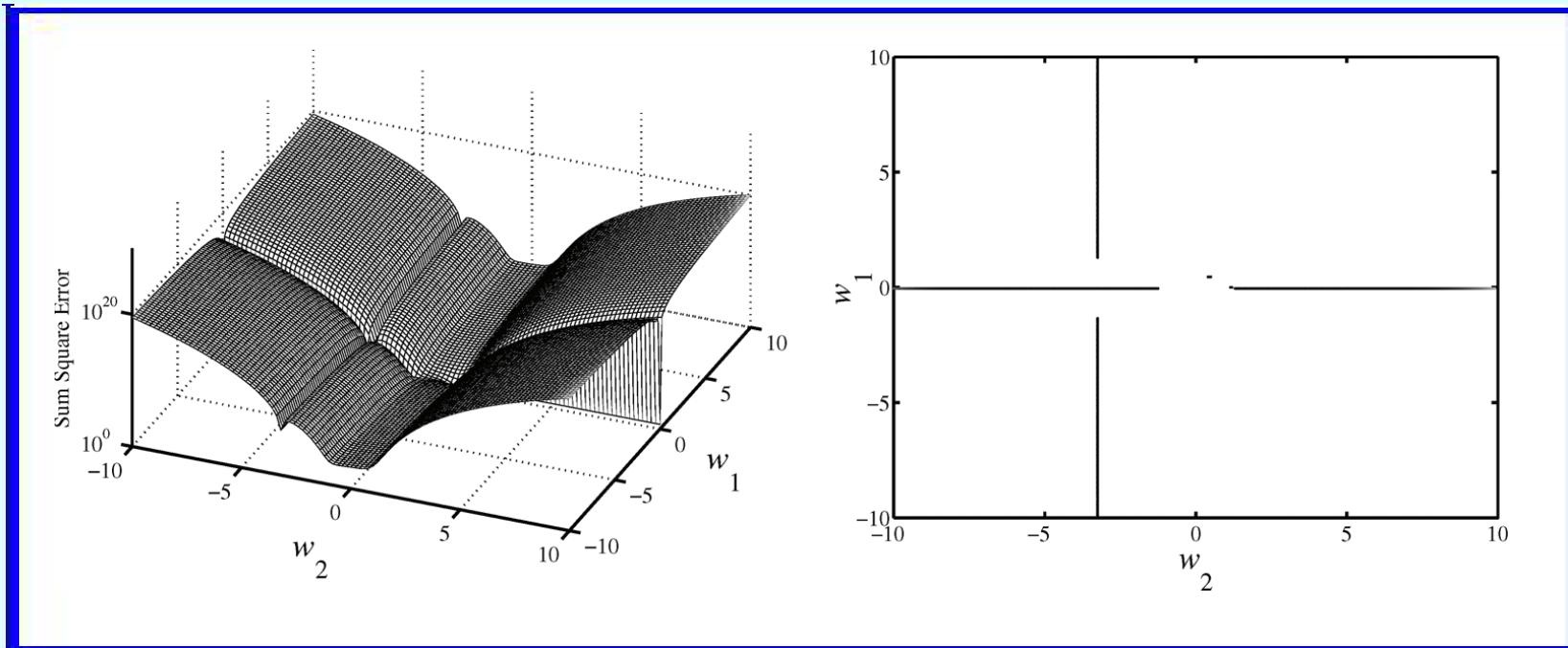
.....

$$k = n : a(n) = w_1 \{p(n) + w_2 p(n - 1) + \dots + w_2^{n-1} p(1)\} + w_2^n a(0)$$

Проблемы при обучении динамических НС-моделей (Х)

Сложный рельеф функции ошибки – 6

Линейная рекуррентная сеть первого порядка (4)

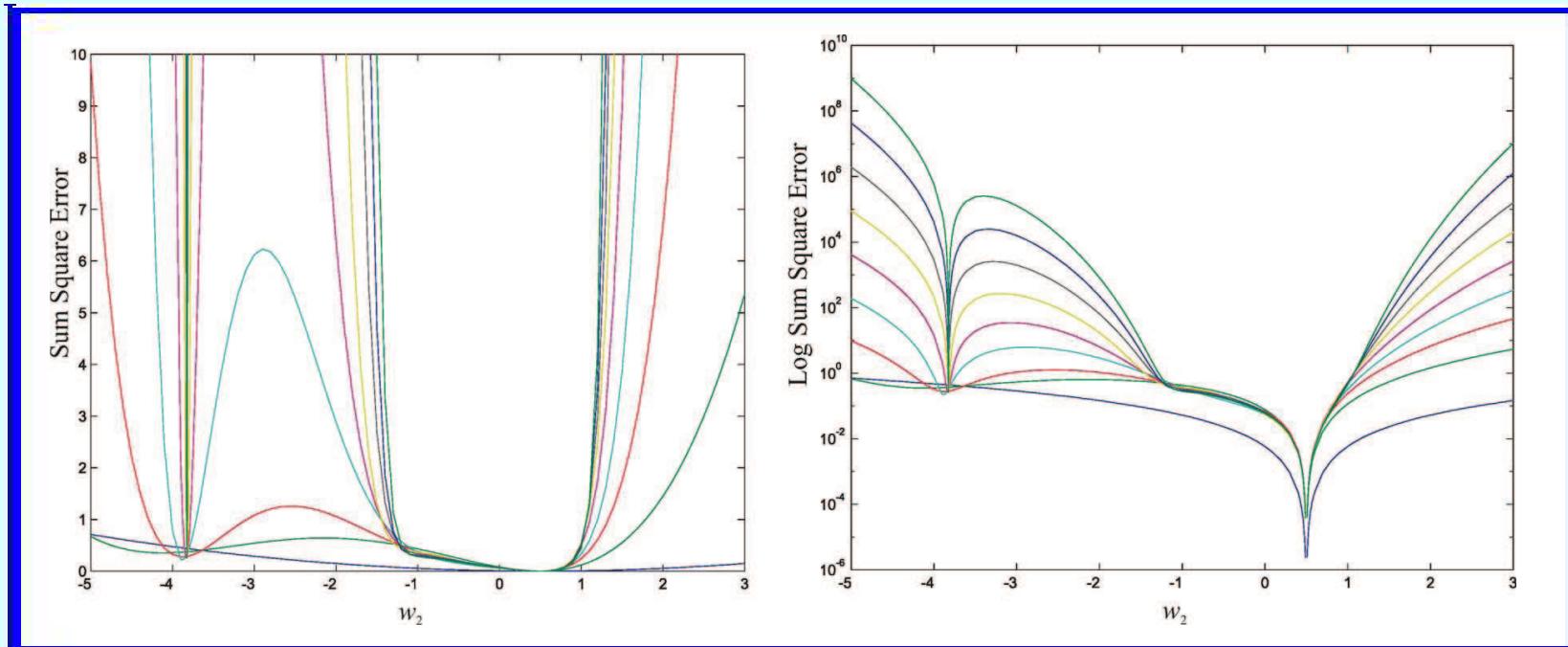


Поверхность ошибки (логарифмический масштаб) и расположение впадин

Источник: *Horn J., De Jesús O., Hagan M.T.* Spurious valleys in the error surface of recurrent networks — analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Проблемы при обучении динамических НС-моделей (XI) Сложный рельеф функции ошибки – 7

Линейная рекуррентная сеть первого порядка (5)



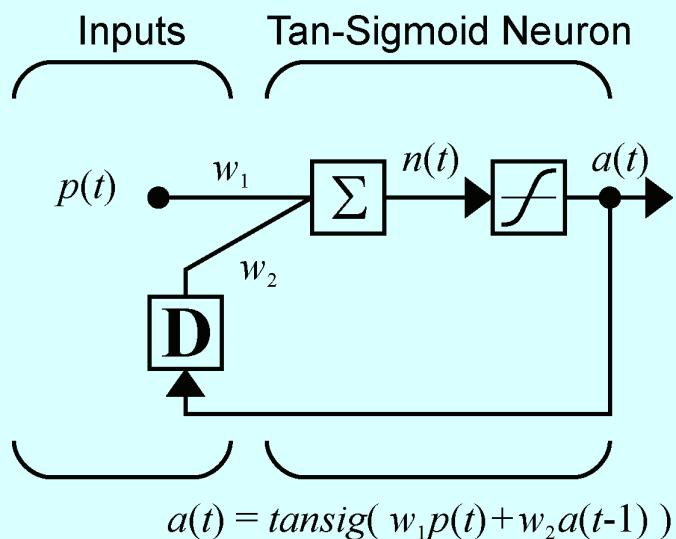
Сечения квадратической функции ошибки при $w_1 = 0.5$
в зависимости от длины последовательности k

Источник: *Horn J., De Jesús O., Hagan M.T.* Spurious valleys in the error surface of recurrent networks — analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Проблемы при обучения динамических НС-моделей (XII)

Сложный рельеф функции ошибки – 8

Нелинейная рекуррентная сеть первого порядка (1)



Формирование обучающего набора: путем подачи на вход сети случайной последовательности $p(t)$ (гауссовский белый шум с нулевым средним и единичной дисперсией), порождающей последовательность выходов $a(t)$.

Значения весов: $w_1 = 0.5$, $w_2 = 0.5$.

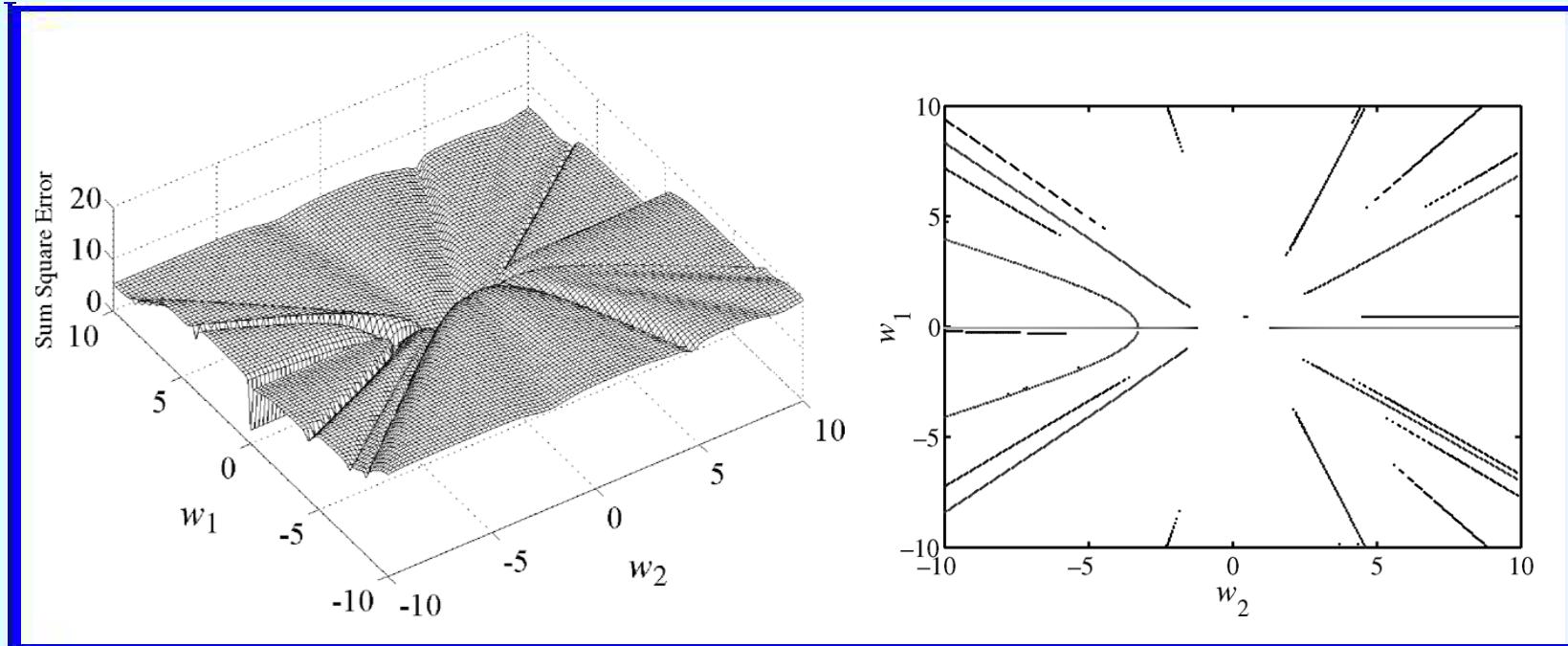
Цель обучения: используя полученный обучающий набор $(p(t), a(t))$, обучить сеть для случайных стартовых значений весов w_1, w_2 .

Отличие этой сети **от предыдущей** только в виде активационной функции — **сигмоида вместо линейной.**

Источник: *Horn J., De Jesús O., Hagan M.T.* Spurious valleys in the error surface of recurrent networks — analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Проблемы при обучении динамических НС-моделей (XIII) Сложный рельеф функции ошибки – 9

Нелинейная рекуррентная сеть первого порядка (2)



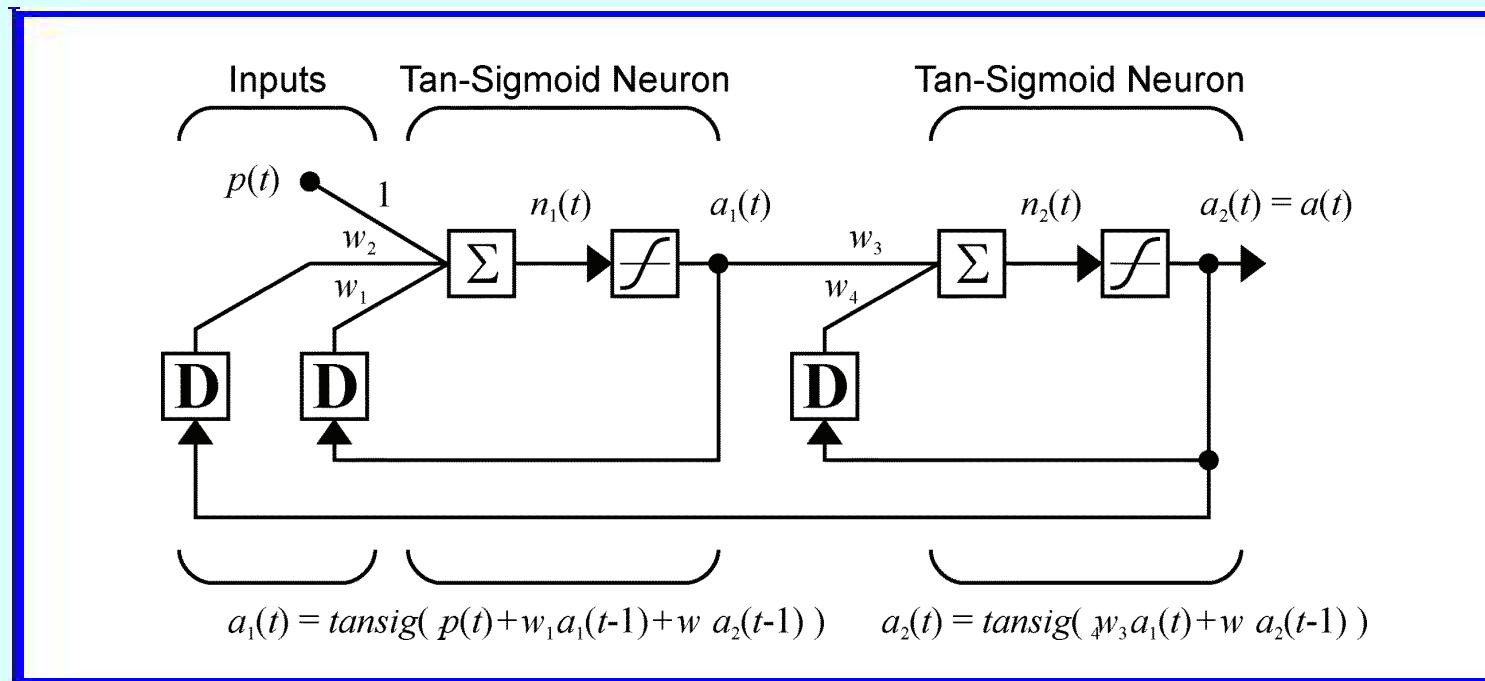
Поверхность ошибки и расположение впадин для **нелинейной сети**

Источник: *Horn J., De Jesús O., Hagan M.T.* Spurious valleys in the error surface of recurrent networks — analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Проблемы при обучении динамических НС-моделей (XIV)

Сложный рельеф функции ошибки – 10

Двухслойная нелинейная рекуррентная сеть (1)

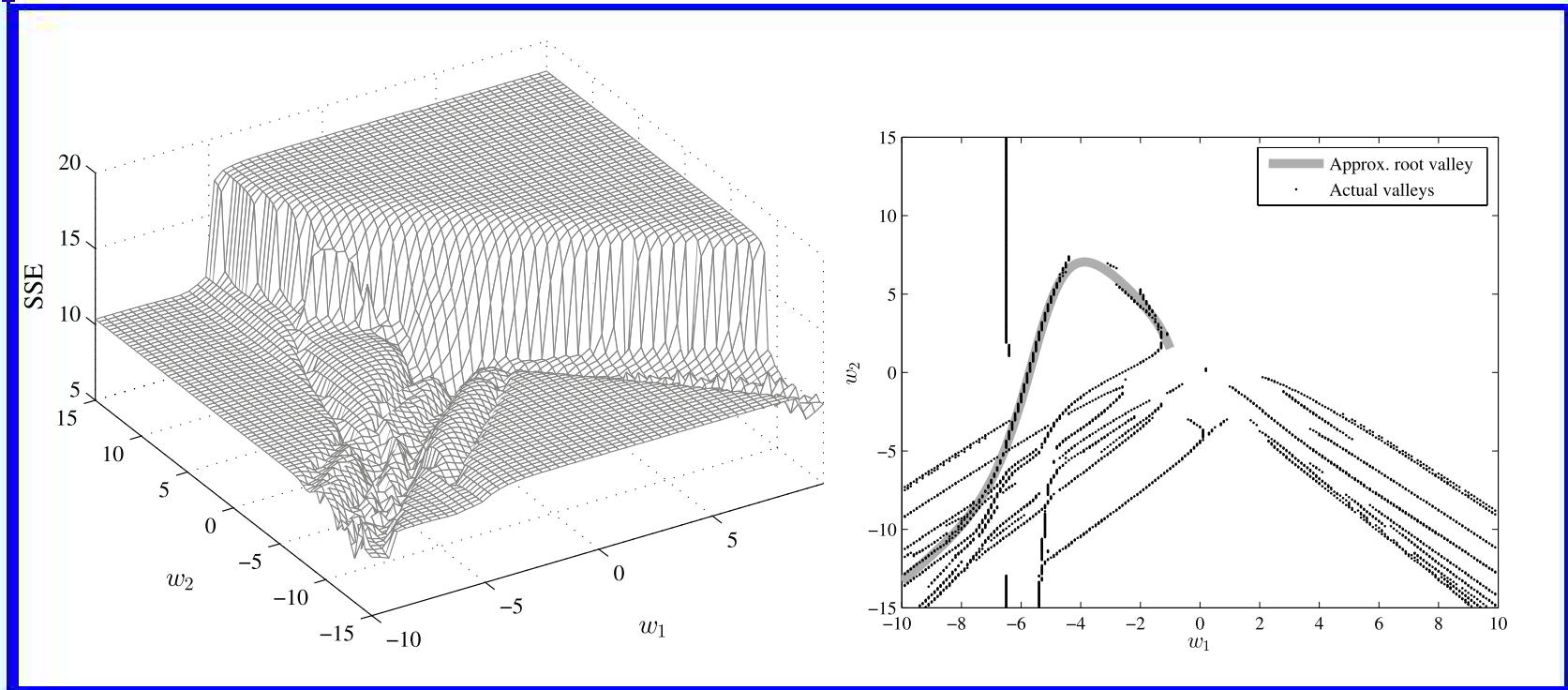


Источник: *Horn J., De Jesús O., Hagan M.T.* Spurious valleys in the error surface of recurrent networks — analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Проблемы при обучении динамических НС-моделей (XIV)

Сложный рельеф функции ошибки – 11

Двухслойная нелинейная рекуррентная сеть (2)



Поверхность ошибки и расположение впадин для двухслойной нелинейной сети

Источник: *Phan M.C., Hagan M.T.* Error surface of recurrent neural networks // IEEE Transactions on Neural Networks. – 2013. – Vol.24, No.11. – pp.1709–1721.

Проблемы при обучения динамических НС-моделей (XV)

Сложный рельеф функции ошибки – 12

Способы преодоления сложного рельефа функции ошибки:

- регуляризация**

$$J(\mathbf{w}) = SSE + \alpha \cdot SSW,$$

где SSE – суммарная среднеквадратическая ошибка сети, SSW – сумма квадратов весов;

- сегментирование** обучающей последовательности (изменение входных данных сети меняет расположение впадин на рельефе функции ошибки);
- случайное варьирование** стартовых значений весов;
- сочетание** регулярного и генетического поиска.

Подробнее см.:

Horn J., De Jesús O., Hagan M.T. Spurious valleys in the error surface of recurrent networks – analysis and avoidance // IEEE Transactions on Neural Networks. – 2009. – Vol.20, No.4. – pp.686–700.

Phan M.C., Hagan M.T. Error surface of recurrent neural networks // IEEE Transactions on Neural Networks. – 2013. – Vol.24, No.11. – pp.1709–1721.

Алгоритмы обучения динамических НС-моделей (I)

Основные классы алгоритмов обучения динамических сетей – 1

Статические НС-модели VS Динамические НС-модели

Статические сети: отсутствуют обратные связи и задержки, выход таких сетей вычисляется непосредственно по их входам с помощью вычислений «распространяющихся вперед», от входов к выходам.

Динамические сети: выход зависит не только от текущего входа сети, но и от ее предшествующих (по времени) входов, выходов и/или состояний.

Динамические сети:

- рекуррентные сети с обратными связями в них;
- сети прямого распространения с линиями задержки в них;
- рекуррентные сети с обратными связями и линиями задержки.

Алгоритмы обучения динамических НС-моделей (II)

Основные классы алгоритмов обучения динамических сетей – 2

В статических НС-моделях вычисление градиента можно выполнить, используя один из вариантов стандартного метода обратного распространения ошибки.

Для динамических НС-моделей приходится использовать **более сложные алгоритмы** динамического вычисления производных, составляющих градиент.

Градиентные алгоритмы обучения рекуррентных сетей:

- обратное распространение во времени (**BPTT** — Back Propagation Through Time);
- рекуррентное обучение в реальном времени (**RTRL** — Real-Time Recurrent Learning);
- расширенный фильтр Калмана (**EKF** — Extended Kalman Filter).

Алгоритмы обучения динамических НС-моделей (III)

Основные классы алгоритмов обучения динамических сетей – 3

Обратное распространение во времени – ВРТТ (1)

В алгоритме ВРТТ отклик НС-модели вычисляется для всего набора моментов времени, в которые фиксировались данные, включаемые в обучающий набор. Затем производится вычисление градиента, начиная с завершающего момента времени и далее назад, в обратном времени, по направлению к входу сети.

Алгоритм ВРТТ достаточно эффективен в вычислительном плане, однако его затруднительно использовать в оперативном (on-line) режиме (надо вначале добраться до завершающего момента времени, что не всегда возможно по смыслу решаемой задачи).

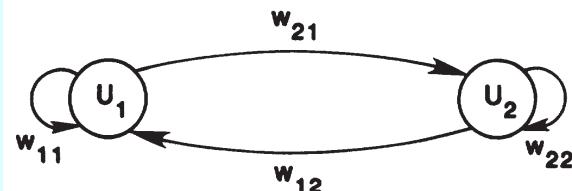
Основная идея алгоритма ВРТТ: любую рекуррентную сеть можно представить в виде эквивалентной ей по поведению слоистой сети прямого распространения с числом слоев, равным числу временных тактов работы рекуррентной сети.

Алгоритмы обучения динамических НС-моделей (IV)

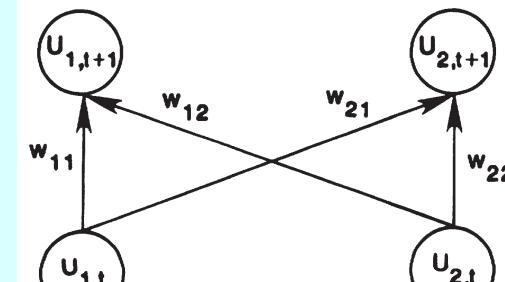
Основные классы алгоритмов обучения динамических сетей – 4

Обратное распространение во времени – ВРТТ (2)

A



B

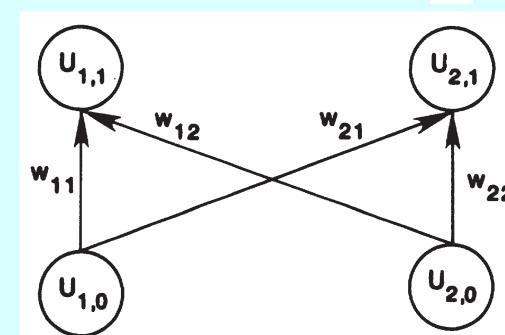


Пример развертывания рекуррентной сети А в эквивалентную ей по поведению многослойную сеть В прямого распространения

А — полносвязная сеть из двух элементов.
В — сеть прямого распространения с поведением таким же, как и у сети А.

Сеть В:

- 1) веса связей от слоя к слою не меняются;
- 2) значения одноименных весов в сетях А и В одни и те же.



Источник: Rumelhart D.E. ao. Learning internal representations by error propagation // In: "Parallel Distributed Processing" / Ed. by D.E.Rumelhart and J.L.McClelland. Vol.1, 1986, p.355.

Алгоритмы обучения динамических НС-моделей (V)

Основные классы алгоритмов обучения динамических сетей – 5

Обратное распространение во времени – ВРТТ (3)

Ошибка сети на отрезке времени от n_0 до n_1 :

$$E(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j \in A} e_j^2,$$

где A — множество индексов j тех нейронов, для которых заданы желаемые отклики; $e_j(n)$ — сигнал ошибки для этих нейронов.

Невязка j -го нейрона (для всех $j \in A$):

$$\delta_j(n) = -\frac{\partial E(n_0, n_1)}{\partial v_j(n)},$$

$v_j(n)$ — выход сумматора j -го нейрона, $\varphi(v_j(n))$ — активационная функция,

$$\delta_j(n) = \begin{cases} \varphi'(v_j(n))e_j(n), & n = n_1; \\ \varphi'(v_j(n)) [e_j(n) + \sum_{k \in A} w_{jk}\delta_k(n+1)], & n_0 < n < n_1. \end{cases}$$

Вычисления для $\delta_j(n)$ повторяются с момента времени n_1 , шаг за шагом, пока не будет достигнут момент времени n_0 .

Алгоритмы обучения динамических НС-моделей (VI) Основные классы алгоритмов обучения динамических сетей – 6

Обратное распространение во времени – ВРТТ (4)

После вычисления невязок $\delta_j(n)$ всех нейронов $j \in A$ можно найти корректировки синаптических весов w_{ji} :

$$\Delta w_{ji} = -\eta \frac{\partial E(n_0, n_1)}{\partial w_{ji}} = \eta \sum_{n=n_0+1}^{n_1} \delta_j(n) x_i(n-1),$$

где η – скорость обучения; $x_i(n-1)$ – входной сигнал, поданный на i -й синапс j -го нейрона в момент времени $n-1$.

Сравнение алгоритма ВРТТ со стандартным алгоритмом ВР показывает, что **главное различие** между ними состоит в том, что для ВРТТ **известны желаемые отклики** для всех слоев эквивалентной сети прямого распространения, так как эти слои получены дублированием выходного слоя исходной рекуррентной сети.

См. также: *Хайкин С.* Нейронные сети: Полный курс. – М.: Вильямс, 2006. – с.943–949.

Werbos P.J. Backpropagation through time: What it does and how to do it // Proceedings of the IEEE. – 1990. Vol.78, No.10. – pp.1550–1560.

Алгоритмы обучения динамических НС-моделей (VII)

Основные классы алгоритмов обучения динамических сетей – 7

Рекуррентное обучение в реальном времени – RTRL (1)

В алгоритме ВРТТ вычисление градиента производится, начиная **с завершающего момента времени и далее назад**, в обратном времени, по направлению к входу сети, т.е. для вычисления градиента надо вначале добраться до завершающего момента времени.

В отличие от этого, **в алгоритме RTRL** градиент может быть вычислен **в тот же самый момент времени**, когда получена реакция НС-модели, т.е. его стартовой точкой является начальный момент времени, а затем используется распространение вперед (в прямом времени).

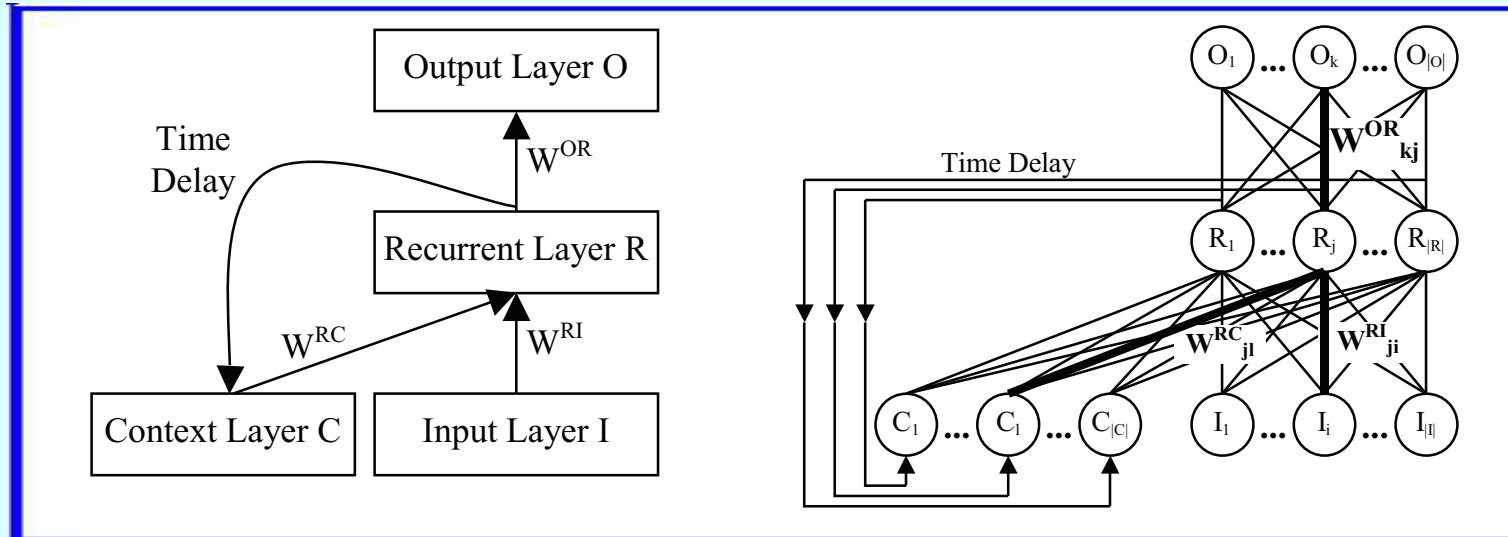
Алгоритм RTRL **требует большего объема вычислений**, чем алгоритм ВРТТ, однако он пригоден для обучения НС-модели **в оперативном режиме (on-line)**.

В то же время, **для вычисления якобиана RTRL в общем случае более эффективен, чем ВРТТ** за счет того, что в RTRL вычисление якобиана является составной частью процесса вычисления градиента.

Алгоритмы обучения динамических НС-моделей (VIII)

Основные классы алгоритмов обучения динамических сетей – 8

Рекуррентное обучение в реальном времени – RTRL (2)



Сеть Элмана

I — входной слой, **R** — рекуррентный (скрытый) слой,

O — выходной слой, **C** — контекстный слой

W^{RI} , W^{RC} , W^{OR} — матрицы весов связей между входным и рекуррентным, контекстным и рекуррентным, рекуррентным и выходным слоями, соответственно

Источник: Čerňanský M., Beňušková Ľ. Simple recurrent network trained by RTRL and extended Kalman filter algorithms // Neural Network World. – 2003. – Vol.13, No.3. – pp.223–234.

См. также: **Elman J.L** Finding structure in time // Cognitive Science. – 1990. – Vol.14, No.2. – pp.179–211.

Алгоритмы обучения динамических НС-моделей (IX)

Основные классы алгоритмов обучения динамических сетей – 9

Рекуррентное обучение в реальном времени – RTRL (3)

Сеть Элмана

Паттерны для момента времени t :

$I^{(t)} = (I_1^{(t)}, \dots, I_j^{(t)}, \dots, I_{NI}^{(t)})$ — во входном слое,

$R^{(t)} = (R_1^{(t)}, \dots, R_j^{(t)}, \dots, R_{NR}^{(t)})$ — в рекуррентном слое,

$O^{(t)} = (O_1^{(t)}, \dots, O_j^{(t)}, \dots, O_{NO}^{(t)})$ — в выходном слое,

$C^{(t)} = (C_1^{(t)}, \dots, C_j^{(t)}, \dots, C_{NC}^{(t)})$ — в контекстном слое.

Выходная активность элементов рекуррентного слоя:

$$\tilde{R}_j^{(t)} = \sum_j W_{ij}^{RI} I_j^{(t)} + \sum_j W_{ij}^{RC} R_j^{(t-1)}, \quad \textcolor{red}{R}_j^{(t)} = f(\tilde{R}_j^{(t)}).$$

Выходная активность элементов выходного слоя:

$$\tilde{O}_j^{(t)} = \sum_j W_{ij}^{OR} R_j^{(t)}, \quad \textcolor{red}{O}_j^{(t)} = f(\tilde{O}_j^{(t)}).$$

Алгоритмы обучения динамических НС-моделей (Х)

Основные классы алгоритмов обучения динамических сетей – 10

Рекуррентное обучение в реальном времени – RTRL (4)

Алгоритм RTRL для сети Элмана (1)

Корректировка весов связей для момента времени t :

Веса связей между рекуррентным и выходным слоями:

$$\Delta W_{ij}^{OR} = \alpha (D_i^{(t)} - O_i^{(t)}) f'(\tilde{O}_i^{(t)}) R_j^{(t)},$$

$D^{(t)} = (D_1^{(t)}, \dots, D_j^{(t)}, \dots, D_{|O|}^{(t)})$ – желаемые значения выходов

Веса связей между входным и рекуррентным слоями:

$$\Delta W_{ji}^{RI} = \alpha \sum_k^{|O|} \left[(D_k^{(t)} - O_k^{(t)}) f'(\tilde{O}_k^{(t)}) \sum_{h=1}^{|R|} W_{kh}^{RC} \frac{\partial R_h^{(t)}}{\partial W_{ji}^{RI}} \right],$$

$$\frac{\partial R_h^{(t)}}{\partial W_{ji}^{RI}} = f'(\tilde{R}_i^{(t)}) \left[I_i^{(t)} \delta_{hj}^{kron} + \sum_{l=1}^{|R|} W_{hl}^{RC} \frac{\partial R_l^{(t-1)}}{\partial W_{ji}^{RI}} \right].$$

$|I|, |R|, |O|$ – количество элементов в соответствующих слоях

$$\delta_{hj}^{kron} = 1 \text{ для } h = i, \delta_{hj}^{kron} = 0 \text{ для } h \neq i$$

Алгоритмы обучения динамических НС-моделей (XI) Основные классы алгоритмов обучения динамических сетей – 11

Рекуррентное обучение в реальном времени – RTRL (5)

Алгоритм RTRL для сети Элмана (2)

Корректировка весов связей для момента времени t :

Веса связей между контекстным и рекуррентным слоями:

$$\Delta W_{ji}^{RC} = \alpha \sum_k^{|O|} \left[(D_k^{(t)} - O_k^{(t)}) f'(\tilde{O}_k^{(t)}) \sum_{h=1}^{|R|} W_{kh}^{RC} \frac{\partial R_h^{(t)}}{\partial W_{ji}^{RC}} \right],$$
$$\frac{\partial R_h^{(t)}}{\partial W_{ji}^{RC}} = f'(\tilde{R}_i^{(t)}) \left[R_i^{(t-1)} \delta_{hj}^{kron} + \sum_{l=1}^{|R|} W_{hl}^{RC} \frac{\partial R_l^{(t-1)}}{\partial W_{ji}^{RC}} \right].$$

Алгоритм RTRL может использоваться для обучения широкого класса динамических сетей, в том числе на основе архитектуры **ЛДДН** (Layered Digital Dynamic Network).

См. также: *Хайкин С.* Нейронные сети: Полный курс. – М.: Вильямс, 2006. – с.949–954.

Williams R.J., Zipser D. A learning algorithm for continually running fully recurrent neural networks // Neural Computation. – 1989. Vol.1, No.2. – pp.270–280.

De Jesús O., Hagan M.T. Backpropagation algorithms for a broad class of dynamic networks // IEEE Transactions on Neural Networks. – 2007. – Vol.18, No.1. – pp.14–27.

Алгоритмы обучения динамических НС-моделей (XII)

Основные классы алгоритмов обучения динамических сетей – 12

Расширенный фильтр Калмана – EKF (1)

Алгоритм стандартного фильтра Калмана:

(линейные системы)

Модель в пространстве состояний:

$$\mathbf{x}_{k+1} = \mathbf{F}_{k+1,k} \mathbf{x}_k + \xi_k,$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \nu_k,$$

Здесь ξ_k и ν_k – гауссовские шумы с нулевым средним и с ковариационными матрицами \mathbf{Q}_k \mathbf{R}_k , соответственно.

Инициализация:

Для $k = 0$ задать

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0],$$

$$\mathbf{P}_0 = E[(\mathbf{x}_0 - E[\mathbf{x}_0])(\mathbf{x}_0 - E[\mathbf{x}_0])^T].$$

Для $k = 1, 2, \dots$ вычислять:

Оценка состояния

$$\hat{\mathbf{x}}_k^- = \mathbf{F}_{k,k-1} \hat{\mathbf{x}}_{k-1}^-;$$

Оценка ковариации ошибки

$$\mathbf{P}_k^- = \mathbf{F}_{k,k-1} \mathbf{P}_{k-1} \mathbf{F}_{k,k-1}^T + \mathbf{Q}_{k-1};$$

Матрица коэффициентов усиления

$$\mathbf{G}_k = \mathbf{P}_k^- \mathbf{H}_k^T [\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k]^{-1};$$

Корректировка оценки состояния

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{G}_k (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-);$$

Корректировка оценки ковариации ошибки

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{G}_k \mathbf{H}_k) \mathbf{P}_k^-.$$

Источник: Kalman filtering and neural networks / Ed. by S.Haykin. –

New York ao: John Wiley & Sons, Inv.– 2001. – p.10.

Алгоритмы обучения динамических НС-моделей (XIII)

Основные классы алгоритмов обучения динамических сетей – 13

Расширенный фильтр Калмана – EKF (2)

Алгоритм расширенного фильтра Калмана: (нелинейные системы)

Модель в пространстве состояний:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(k, \mathbf{x}_k) + \xi_k, \\ \mathbf{y}_k &= \mathbf{h}(k, \mathbf{x}_k) + \nu_k, \end{aligned}$$

Здесь ξ_k и ν_k — гауссовские шумы с нулевым средним и с ковариационными матрицами \mathbf{Q}_k \mathbf{R}_k , соответственно.

Определения:

$$\begin{aligned} \mathbf{F}_{k+1,k} &= \frac{\partial \mathbf{f}(k, \mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_0}, \\ \mathbf{H}_k &= \frac{\partial \mathbf{h}(k, \mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_0}. \end{aligned}$$

Инициализация:

Для $k = 0$ задать
 $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$,

$$\mathbf{P}_0 = E[(\mathbf{x}_0 - E[\mathbf{x}_0])(\mathbf{x}_0 - E[\mathbf{x}_0])^T].$$

Для $k = 1, 2, \dots$ вычислять:

Оценка состояния

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(k, \hat{\mathbf{x}}_{k-1});$$

Оценка ковариации ошибки

$$\mathbf{P}_k^- = \mathbf{F}_{k,k-1} \mathbf{P}_{k-1} \mathbf{F}_{k,k-1}^T + \mathbf{Q}_{k-1};$$

Матрица коэффициентов усиления

$$\mathbf{G}_k = \mathbf{P}_k^- \mathbf{H}_k^T [\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k]^{-1};$$

Корректировка оценки состояния

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{G}_k \mathbf{y}_k - \mathbf{h}(k, \hat{\mathbf{x}}_k^-);$$

Корректировка оценки ковариации ошибки

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{G}_k \mathbf{H}_k) \mathbf{P}_k^-.$$

Источник: Kalman filtering and neural networks / Ed. by S.Haykin. – New York ao: John Wiley & Sons, Inv.– 2001. – p.19.

Алгоритмы обучения динамических НС-моделей (XIV)

Основные классы алгоритмов обучения динамических сетей – 14

Расширенный фильтр Калмана – EKF (3)

Будем считать, что для идеальной НС-модели наблюдаемый процесс является **стационарным**, т.е. $w_{k+1} = w_k$, но состояния его (веса w_k) «испорчены» шумами ξ_k .

Фильтр Калмана (ФК) в его стандартном варианте применим только для систем, наблюдения которых линейны по оцениваемым параметрам, в то время как уравнение наблюдения нейросети является **нелинейным**:

$$w_{k+1} = w_k + \xi_k,$$

$$\hat{y}_k = f(u_k, w_k) + \nu_k,$$

где u_k — управляющие воздействия, ξ — шум объекта и ν — шум наблюдений, эти шумы являются гауссовскими случайными последовательностями с нулевым средним и ковариационными матрицами Q и R .

Алгоритмы обучения динамических НС-моделей (XV)

Основные классы алгоритмов обучения динамических сетей – 15

Расширенный фильтр Калмана – EKF (4)

Для того, чтобы использовать ФК, требуется линеаризовать уравнение **наблюдения**. Можно использовать **статистическую линеаризацию**, т. е. линеаризацию относительно математического ожидания. Она дает:

$$w_{k+1} = w_k + \xi_k,$$
$$\hat{y}_k = H_k w_k + \nu_k,$$

где **матрица наблюдения** имеет вид

$$H_k = \frac{\partial \hat{y}}{\partial w^T} \Bigg|_{\substack{w=w_k \\ x=x_k}} = - \frac{\partial e_k}{\partial w_k^T} = -J_k.$$

Здесь e_k — вектор ошибки наблюдения на k -м шаге оценивания.

Алгоритмы обучения динамических НС-моделей (XVI)

Основные классы алгоритмов обучения динамических сетей – 16

Расширенный фильтр Калмана – EKF (5)

Уравнения расширенного фильтра Калмана (РФК) для оценки w_{k+1} на следующем шаге имеют вид:

$$\begin{aligned} S_k &= H_k P_k H_k^T + R_k, \\ K_k &= P_k H_k^T S_k^{-1}, \\ P_{k+1} &= (P_k - K_k H_k P_k) e^\beta + Q_k, \\ w_{k+1} &= w_k + K_k e_k, \end{aligned}$$

Здесь e_k – вектор ошибки наблюдения на k -м шаге оценивания:

$$e_k = y_k - \hat{y}_k = y_k - f(x_k, w_k),$$

а величина β – коэффициент забывания, влияет на значимость предыдущих шагов.

Здесь обозначено также: K_k – калмановский коэффициент усиления, S_k – ковариационная матрица ошибок e_k , P_k – ковариационная матрица ошибок оценивания ($\hat{w}_k - w_k$).

Алгоритмы обучения динамических НС-моделей (XVII)

Основные классы алгоритмов обучения динамических сетей – 17

Расширенный фильтр Калмана – EKF (6)

Существуют **альтернативные варианты РФК**, которые могут оказаться более эффективными при решении рассматриваемых задач, в частности, такой:

$$P_k^- = P_k + Q_k,$$

$$S_k = H_k P_k^- H_k^T + R_k,$$

$$K_k = P_k^- H_k^T S_k^{-1},$$

$$P_{k+1} = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k K_k^T,$$

$$w_{k+1} = w_k + K_k e_k.$$

Вариант РФК данного вида **более устойчив** в вычислительном отношении, обладает **робастностью** к ошибкам округления, что положительно влияет на обеспечение вычислительной устойчивости процесса обучения НС-модели в целом.

Алгоритмы обучения динамических НС-моделей (XVIII)

Основные классы алгоритмов обучения динамических сетей – 18

Расширенный фильтр Калмана – EKF (7)

Как видно из соотношений, определяющих РФК, **ключевым моментом** опять является **вычисление якобиана** J_k ошибок сети по настраиваемым параметрам.

При обучении нейросети использовать в РФК только текущее измерение нельзя из-за недопустимо низкой точности определения поиска (влияние шумов ξ и ν), необходимо формировать векторную оценку на интервале наблюдений, тогда обновление матрицы P_k происходит более корректно.

В качестве **вектора наблюдений** можно взять последовательность значений на некотором скользящем интервале:

$$\hat{y}_k = [\hat{y}_{i-l}, \hat{y}_{i-l+1}, \dots, \hat{y}_i]^T,$$

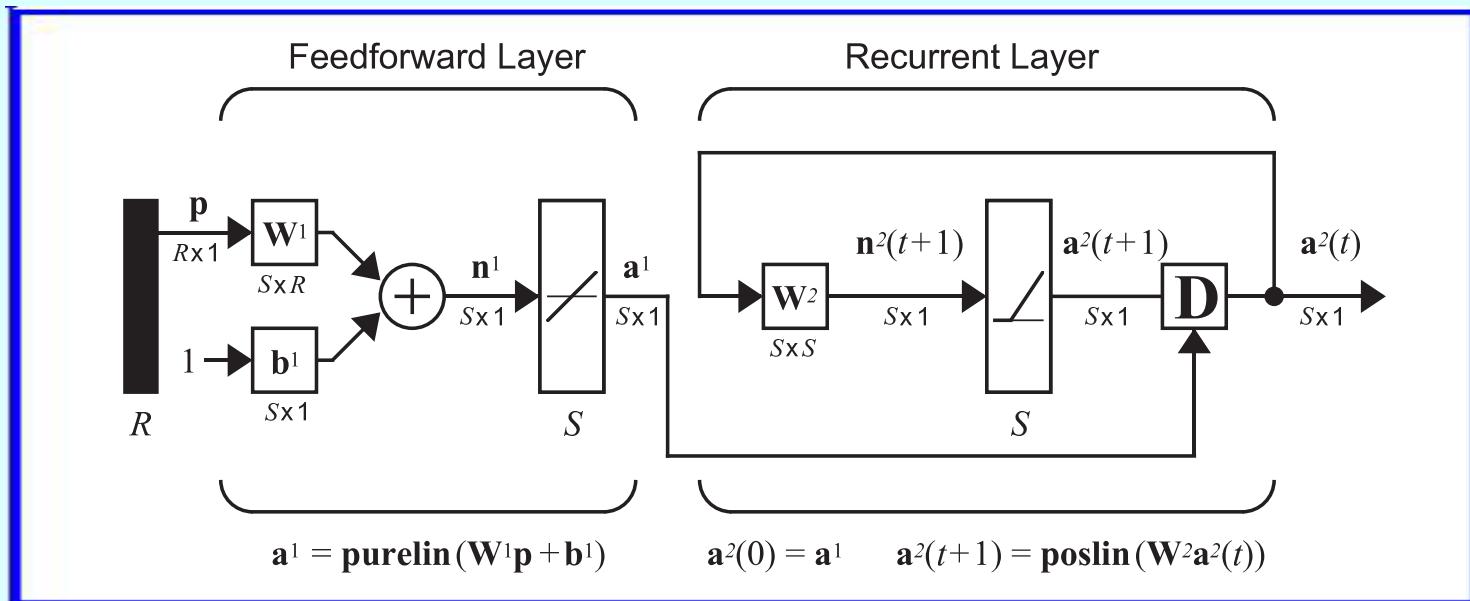
где l — длина скользящего интервала, индекс i относится к моменту времени (шагу дискретизации), а индекс k указывает номер оценки.

Ошибка НС-модели также будет векторной величиной:

$$e_k = [e_{i-l}, e_{i-l+1}, \dots, e_i]^T.$$

Модели с соревновательным обучением (I)

Сеть Хемминга

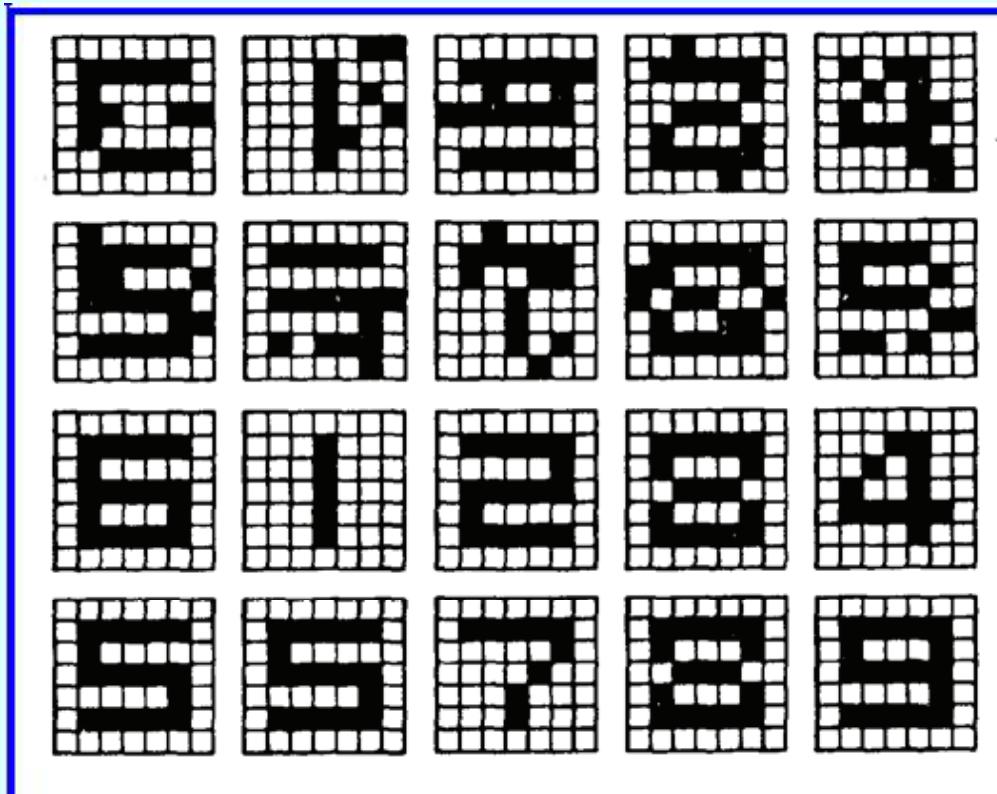


Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 14, Figure 14.1, p.14-3).

Модели с соревновательным обучением (II)

Пример применения сети Хемминга – 1

Распознавание цифр сетью Хемминга (1)



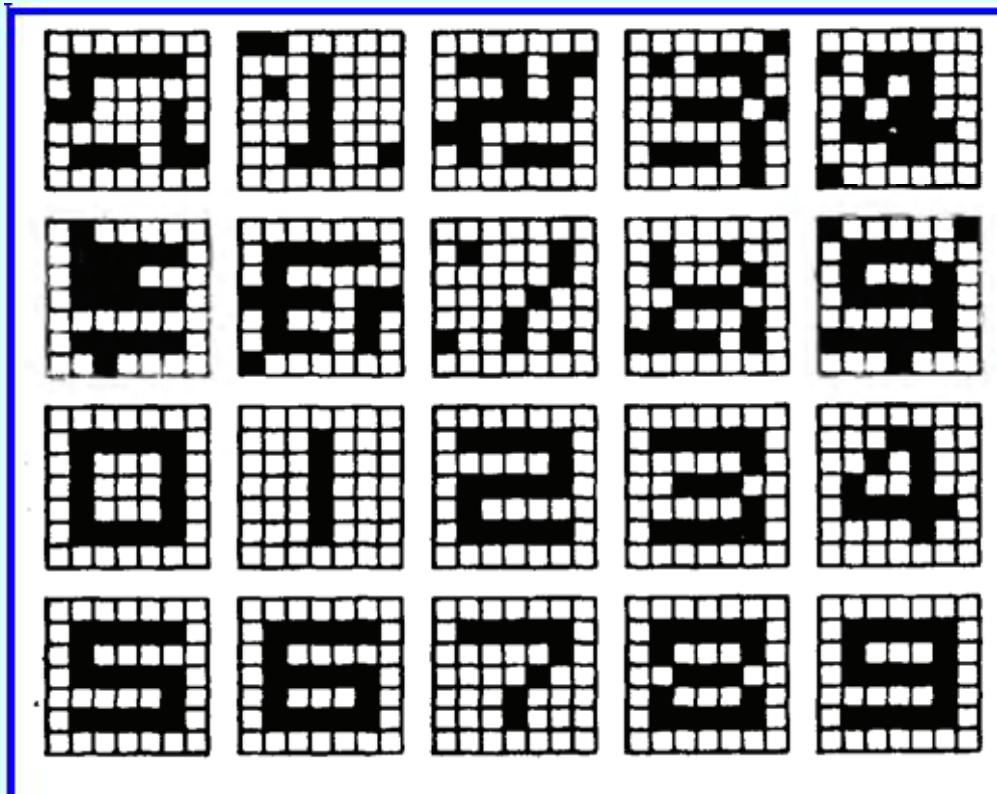
Тестовые (вверху) и распознанные сетью Хемминга (внизу) образы цифр

Источник: [Осовский С.](#) Нейронные сети для обработки информации: Пер. с польск. – М.: Финансы и статистика, 2002. – 344 с. (Глава 7, рис.7.5, с.188).

Модели с соревновательным обучением (III)

Пример применения сети Хемминга – 2

Распознавание цифр сетью Хемминга (2)



Тестовые (вверху) и распознанные сетью Хемминга (внизу) образы цифр

Источник: [Осовский С.](#) Нейронные сети для обработки информации: Пер. с польск. – М.: Финансы и статистика, 2002. – 344 с. (Глава 7, рис.7.6, с.188).

Модели с соревновательным обучением (IV)

Основные подходы к обучению НС

Обучение с учителем: известны векторные входы НС-модели p_j и выходы t_j , отвечающие этим входам, т.е. определено **требуемое поведение** сети в виде обучающего набора $\{\langle p_1, t_1 \rangle, \dots, \langle p_j, t_j \rangle, \dots, \langle p_q, t_q \rangle\}$. **Цель обучения** — так подобрать значения весов и смещений сети, чтобы выходы сети были максимально близки заданным в обучающем наборе t_j . **Типичная задача** — приближенное представление зависимостей.

Обучение без учителя: известны **только** векторные входы НС-модели p_j , значения выходов t_j , отвечающих этим входам, отсутствуют. В этом случае обучающий набор принимает вид $\{p_1, \dots, p_j, \dots, p_q\}$. **Цель обучения** — сгруппировать элементы, входящие в обучающий набор, основываясь на закономерностях, присущих этому набору. **Типичная задача** — классификация (категоризация) данных.

Модели с соревновательным обучением (V)

Обучение без учителя (1)

Обучение без учителя ≡ Объективная классификация

Проблемы реализации алгоритмов объективной классификации:

- выбор **способа разбиения** объектов на классы;
- выработка **правил отнесения** входного образца (паттерна) к определенному классу.

Модели с соревновательным обучением (VI)

Обучение без учителя (2)

Разбиение на классы \Leftarrow **Компактность образов (классов)**

Компактность образов: Каждому образу (классу) в пространстве параметров соответствует **обособленная совокупность точек**, каждая из точек — отдельный объект (паттерн, образец).

Решение задачи классификации: разделение в многомерном пространстве множества точек на части (классы), про которые известно только, что они **изолированы друг от друга**.

Модели с соревновательным обучением (VII)

Классы объектов в двумерном пространстве

Кластеризация (1)

Критерий разбиения на классы

Проблема: найти границу между классами в многомерном пространстве.

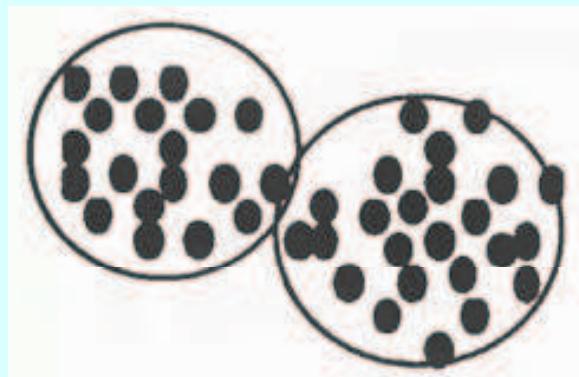
Решается путем использования некоторой меры *близости* или *сходства* объектов.

Мера сходства объектов:

$$d(x_1, x_2) = ||x_1 - x_2|| = \sqrt{\sum_{i=1}^n (\Delta x_i)^2}$$

Объекты — n -мерные векторы x_1 и x_2 .

Мера сходства — евклидово расстояние $d(x_1, x_2)$, как критерий для отнесения объекта к тому или иному классу.



Источник: Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 8.1, с. 204).

Модели с соревновательным обучением (VIII)

Классы объектов в двумерном пространстве

Кластеризация (2)

Нейронная сеть, использующая **алгоритм объективной классификации**, реализует **самообучение** или самоорганизацию — правильные ответы ей не сообщаются.

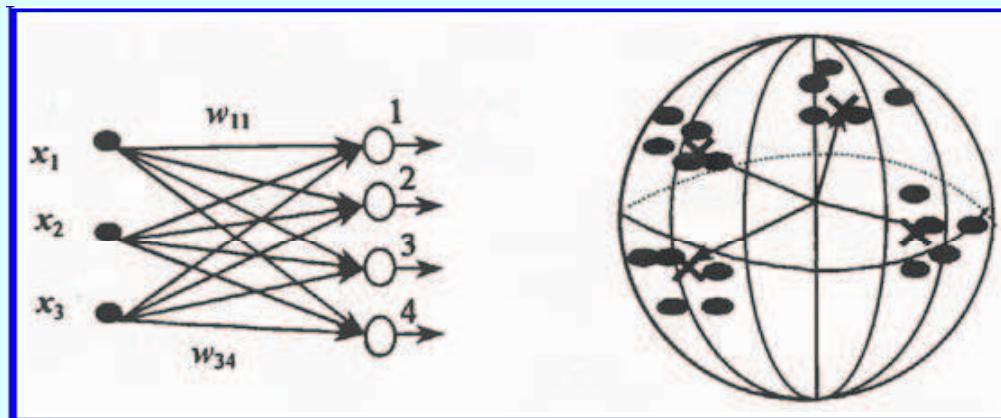
При обучении формируются **кластеры** на основе обобщения входной информации, имеющейся в обучающем наборе.

Обученная сеть относит каждый вновь поступающий входной вектор (паттерн) к одному из кластеров, руководствуясь заданным **критерием сходства**.

Модели с соревновательным обучением (IX)

Конкурентное обучение – 1

Конкурентное (соревновательное) обучение — **наиболее популярная схема** реализации процедуры кластеризации без учителя.



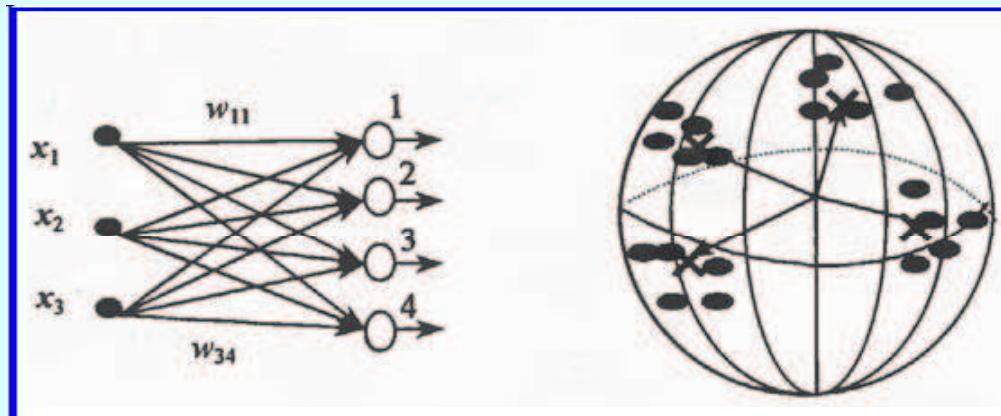
Слева: Пример НС с тремя входами (это размерность *входного вектора*) и четырьмя выходами (это число кластеров, на которое надо разделить входное пространство).

Справа: Кластеризация входных векторов на **сферической поверхности** единичного радиуса. Здесь **X** — центры вновь образующихся кластеров (определяются соответствующим **весовым вектором**).

Источник: *Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры.* Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 8.2, с. 205).

Модели с соревновательным обучением (Х)

Конкурентное обучение – 2



Трехмерное входное пространство разделяется на четыре кластера, кластерные центры изменяются в процессе **конкурентного обучения**.

Входные векторы \mathbf{x} и весовые векторы \mathbf{w}_j , $j = 1, 2, 3, 4$ — нормированные:

$$\|\mathbf{x}\| = \sqrt{\sum_i x_i^2} = 1; \quad \|\mathbf{w}_j\| = \sqrt{\sum_i w_{ij}^2} = 1$$

Источник: Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 8.2, с. 205).

Модели с соревновательным обучением (XI)

Конкурентное обучение – 3

Величина **нейронной активности** a_j для j -го выхода определяется выражением:

$$a_j = \sum_{i=1}^3 x_i w_{ij}$$

Конкуренция между выходами:

Нейрон-победитель определяется по наибольшему значению активности.

Пусть k — номер нейрона-победителя.

Корректировка **весов связей** k -го выхода
(правило «победитель забирает всё»):

$$w_k(t+1) = \frac{w_k(t) + \mu(x_k(t) - w_k(t))}{\|w_k(t) + \mu(x_k(t) - w_k(t))\|}$$

Здесь $0 < \mu < 1$ — скорость обучения.

При предъявлении очередного входного вектора **обновляются только** веса нейрона-победителя, остальные веса не изменяются.

Модели с соревновательным обучением (XII)

Конкурентное обучение – 4

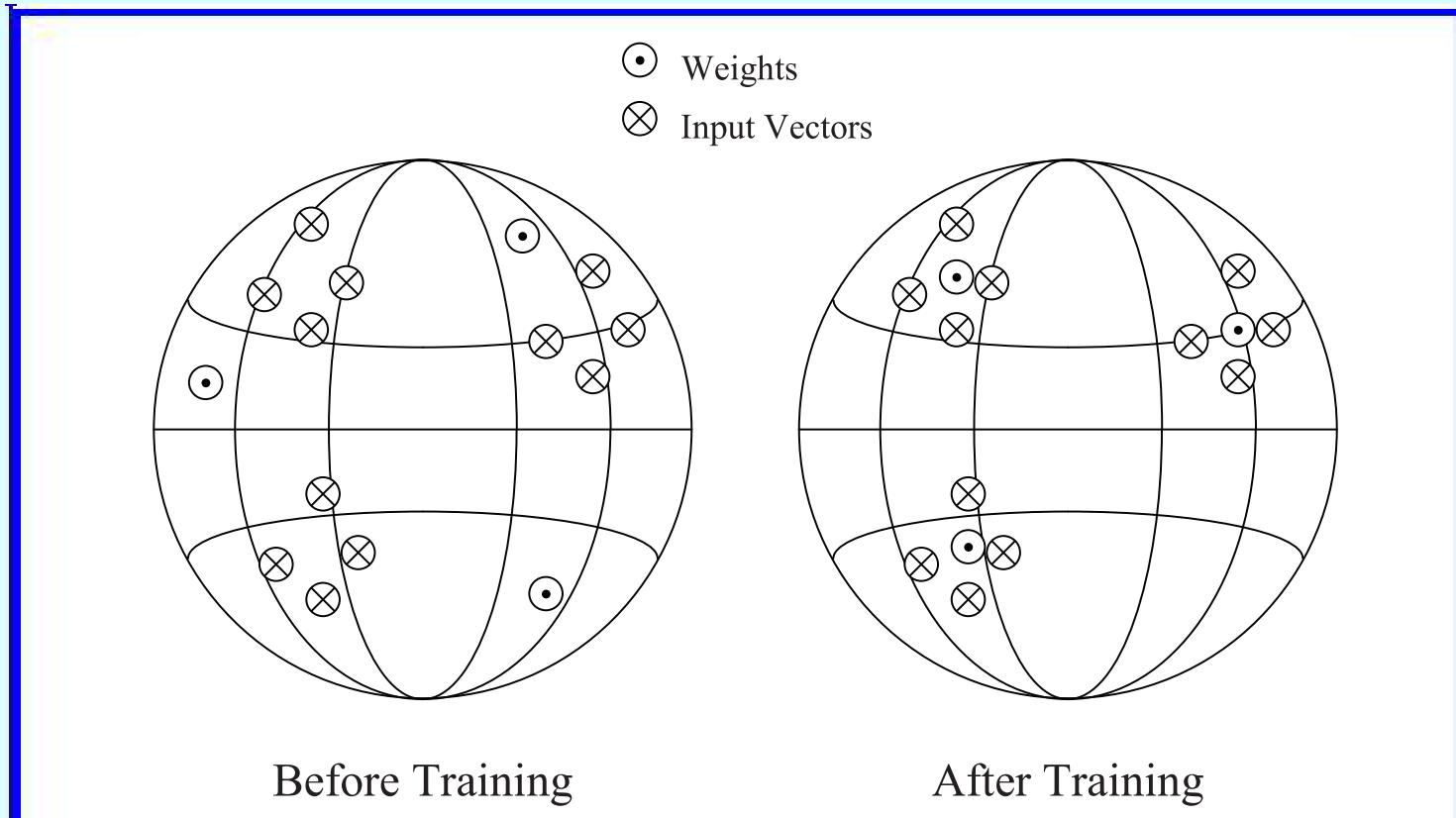
При предъявлении очередного входного вектора **обновляются только** веса нейрона-победителя, остальные веса не изменяются.

Таким образом, в качестве **нейрона-победителя** выбирается такой нейрон, весовой вектор которого **наиболее близок к входному вектору**, именно поэтому весовые векторы выходных нейронов на каждом шаге обучения будет **перемещаться в направлении кластеров**, отвечающих входным образам.

Модели с соревновательным обучением (XIII)

Конкурентное обучение – 5

Сходимость соревновательного процесса кластеризации

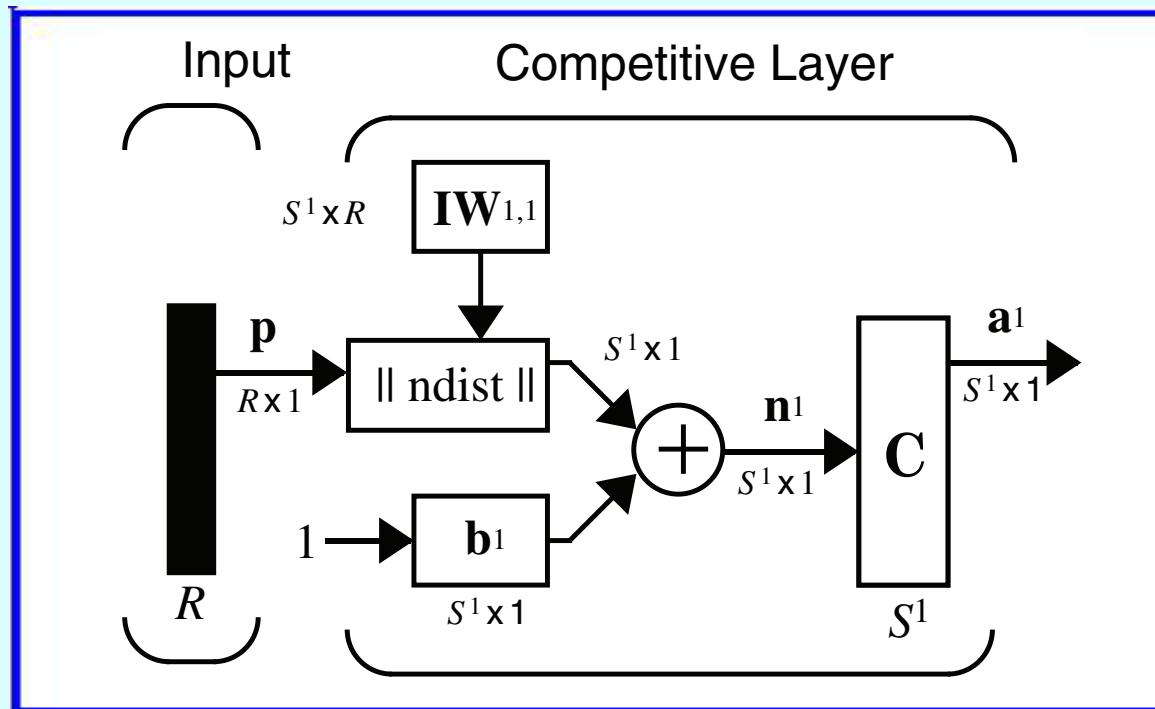


Источник: *Hagan M. T., Demuth H. B., Beale M.*. Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Lecture slides, Part 14, Slide 10).

Модели с соревновательным обучением (XIV)

Конкурентное обучение – 6

Соревновательная сеть

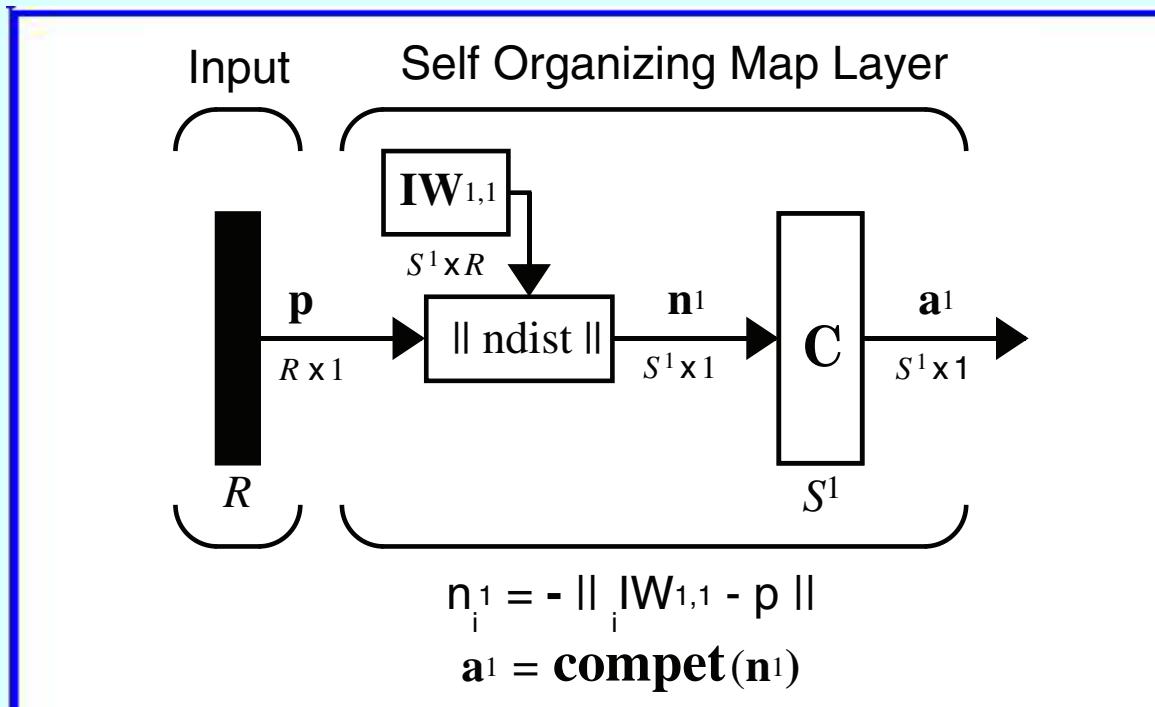


Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 9-3).

Модели с соревновательным обучением (XV)

Самоорганизующиеся карты – 1

SOM – Self-Organizing Map
(SOFM – Self-Organizing Feature Map)



Структурное отличие от соревновательной сети — **нет вектора смещений**.

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 9-17).

Модели с соревновательным обучением (XVI)

Самоорганизующиеся карты – 2

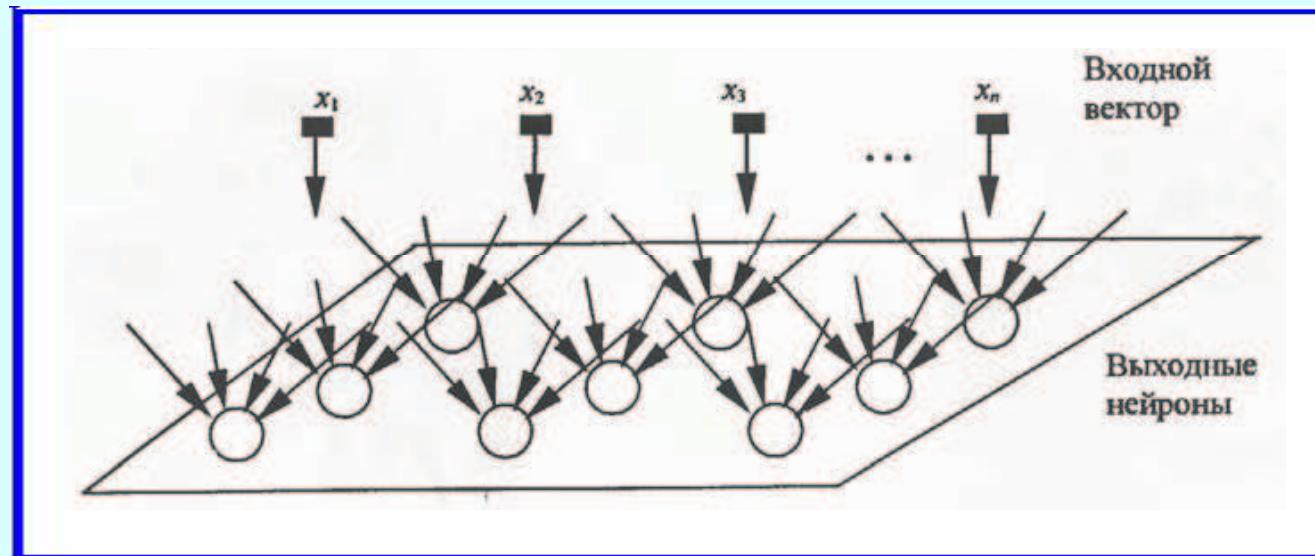
SOM используют для отображения **нелинейных взаимосвязей** многомерных данных на достаточно легко интерпретируемые (чаще всего – двумерные) сетки.

SOM отражают **метрические и топологические зависимости** входных векторов, объединяемых в кластеры.

SOM имеет **один слой** нейронов, **число входов** каждого нейрона равно размерности входного паттерна (образца), **количество нейронов в слое** определяет, сколько различных кластеров сеть сможет распознать.

Модели с соревновательным обучением (XVII)

Самоорганизующиеся карты – 3



SOM:

один слой нейронов

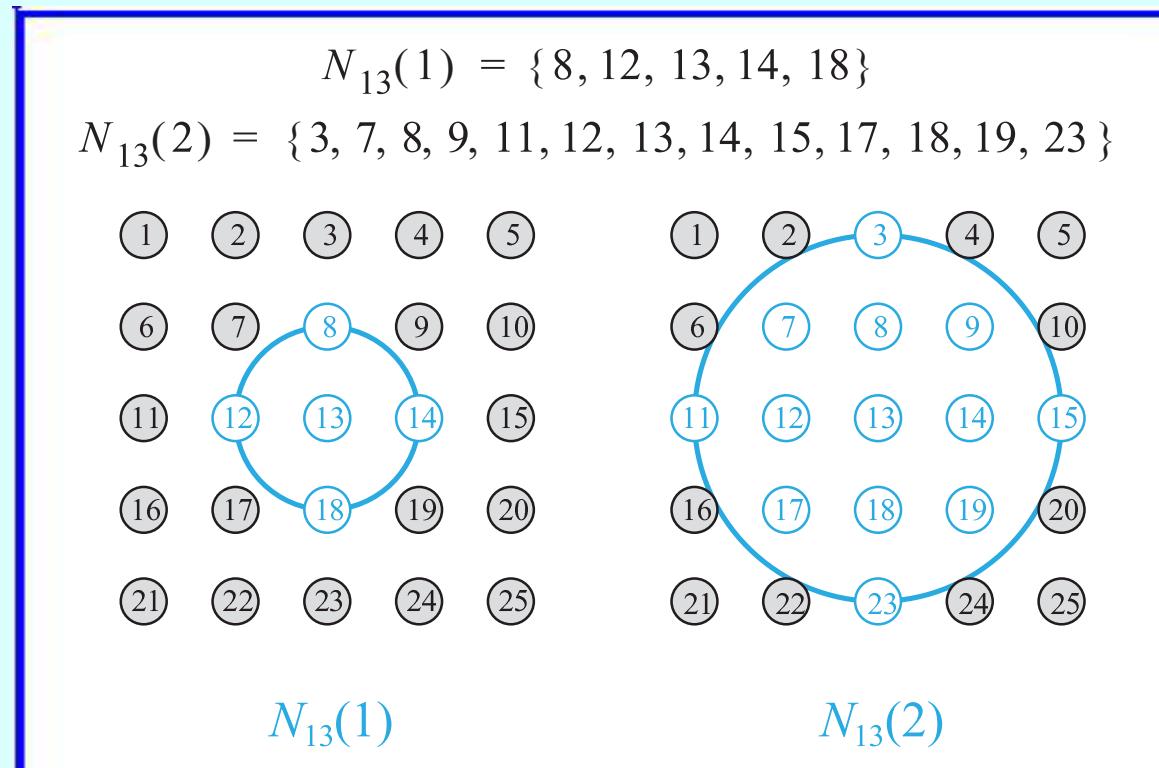
число входов каждого нейрона — размерность входного паттерна
количество нейронов в слое — количество распознаваемых кластеров

Источник: Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 8.3, с. 209).

Модели с соревновательным обучением (XVIII)

Самоорганизующиеся карты – 4

Области соседства



Источник: *Hagan M. T., Demuth H. B., Beale M.* Neural network design. – PWS Publishing Company, 1996. – 730 pp. (Chapter 14, Figure 14.8, p.14-13).

Модели с соревновательным обучением (XIX)

Самоорганизующиеся карты – 5

Латеральные связи в слое



Правило Кохонена

Нейроны в соревновательном слое имеют **связи со своими соседями**.

Модификация весов:
нейрон-победитель + нейроны-соседи

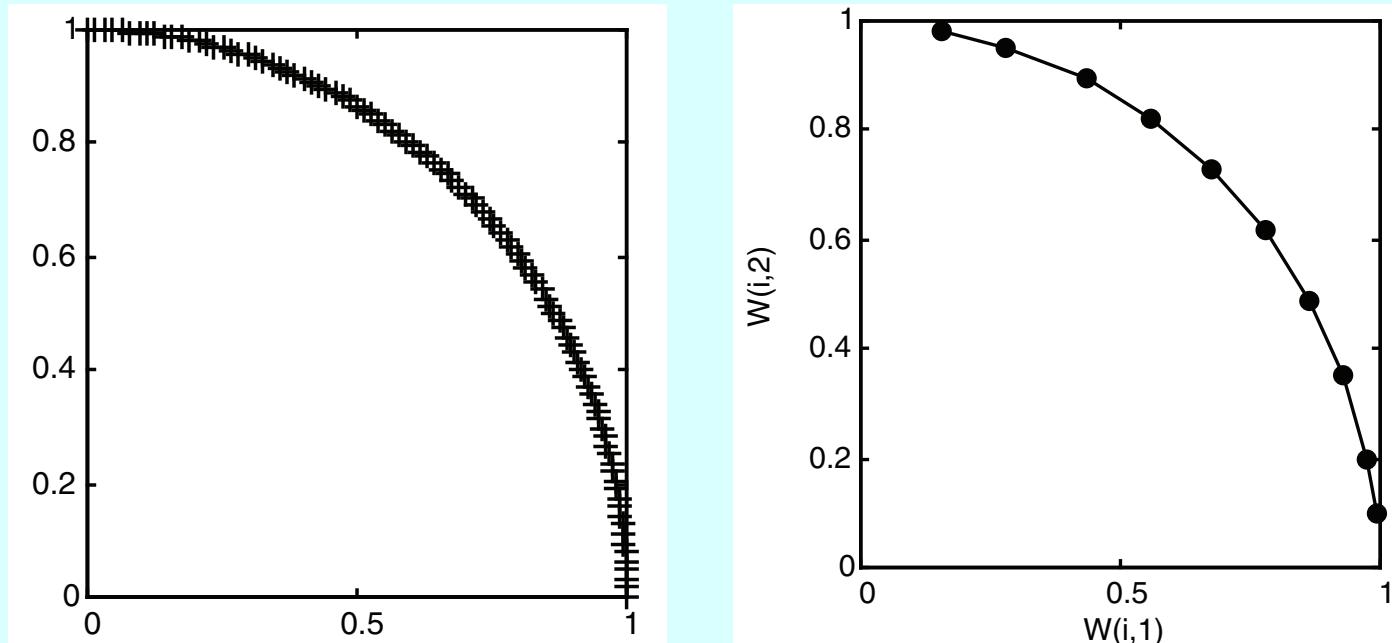
Правило модификации:
 $w_j(t+1) = w_j(t) + \mu \text{dist}(x_j(t), w_j(t))$
 $0 < \mu < 1$ — скорость обучения

Источник: Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 8.9, с. 214).

Модели с соревновательным обучением (ХХ)

Самоорганизующиеся карты – 6

Пример формирования одномерной карты



Входные данные – 100 векторов с компонентами $x_1 = \sin \alpha$, $x_2 = \cos \alpha$, $0 \leq \alpha \leq \pi$ с шагом $0.5 \cdot \pi / 99$.

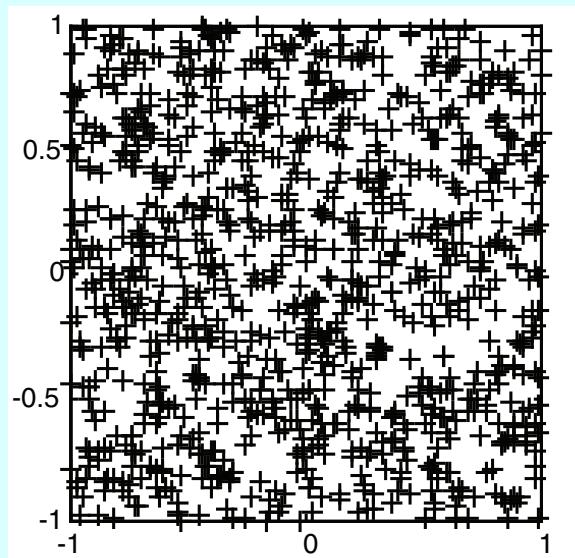
Сеть SOM – одномерный слой из 10 нейронов.

Источник: Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. Изд. 2-е. – М.: Изд-во МГТУ им. Н. Э. Баумана. – 2004. – 400 pp. (рис. 8.9, с. 214).

Модели с соревновательным обучением (ХI)

Самоорганизующиеся карты – 7

Пример формирования двумерной карты



Входные векторы:

1000 случайных векторов $\mathbf{x} = (x_1, x_2)$

Сеть SOM:

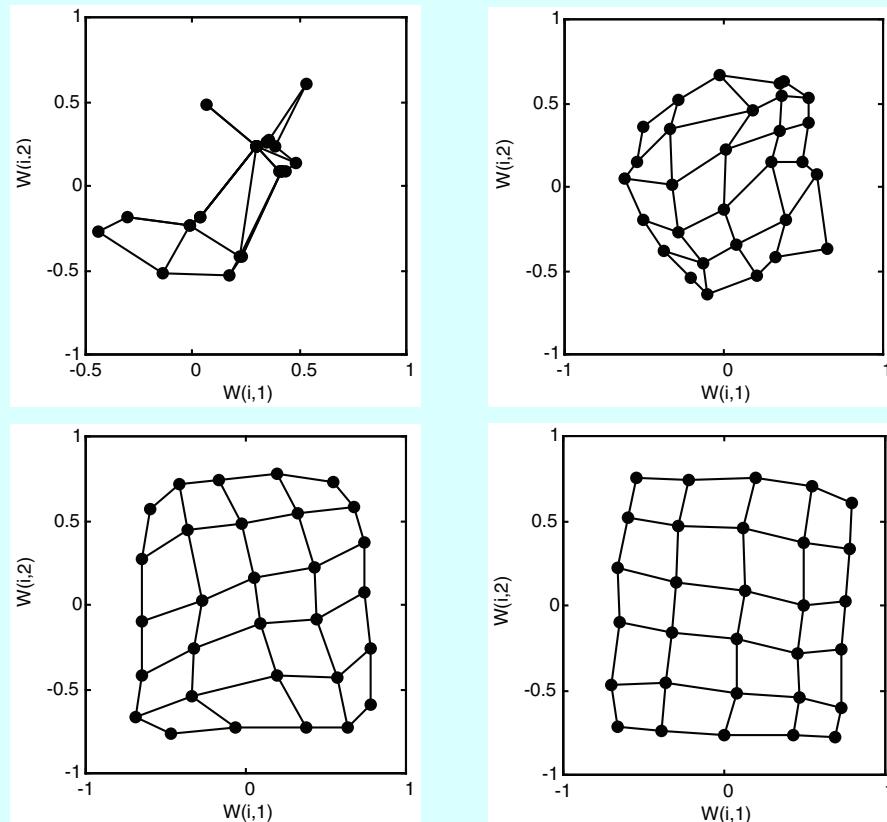
двумерный слой из $5 \times 6 = 30$ нейронов.

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 9-26).

Модели с соревновательным обучением (ХII)

Самоорганизующиеся карты – 8

Пример формирования двумерной карты

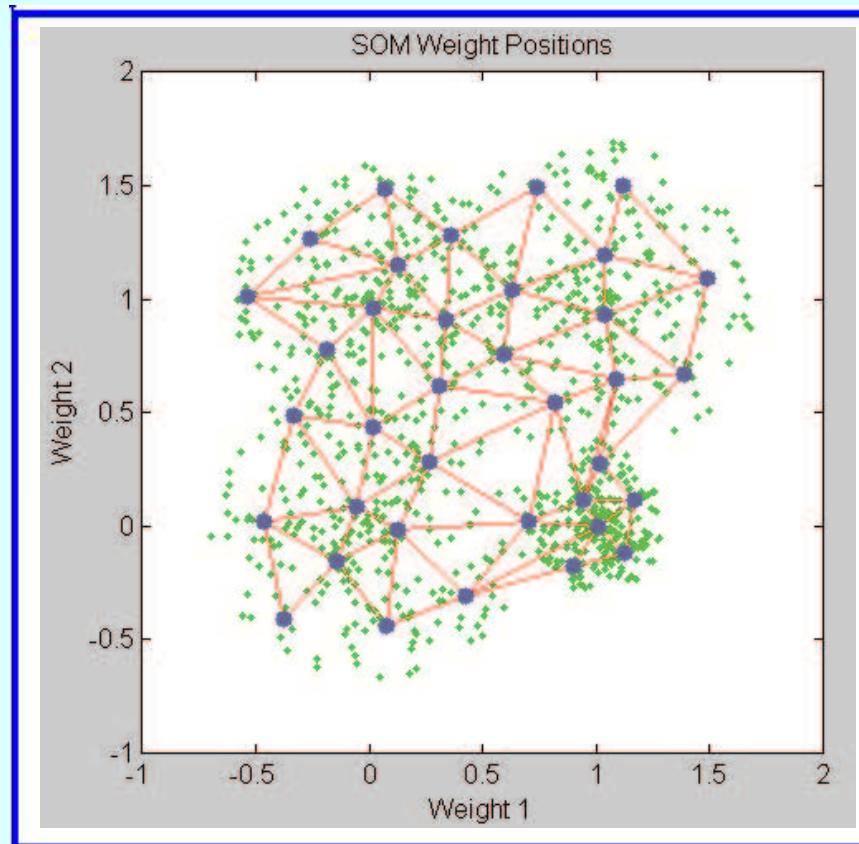


Слева направо, сверху вниз: после 40 эпох; после 120 эпох; после 500 эпох; после 5000 эпох

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (pp. 9-26, 9-27, 9-28).

Модели с соревновательным обучением (ХХIII)

Самоорганизующиеся карты – 9
Пример формирования двумерной карты



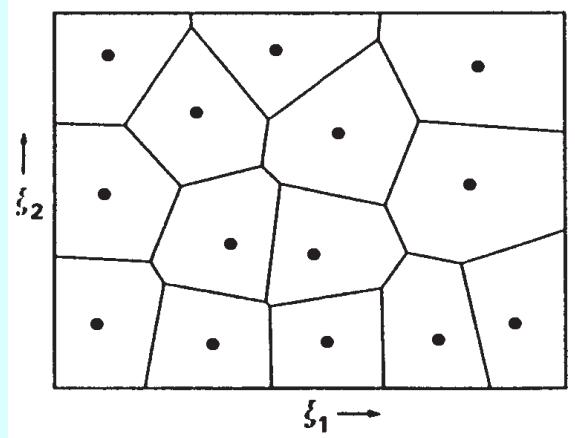
Неоднородное распределение входных векторов, 200 эпох

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (p. 9-30).

Модели с соревновательным обучением (XXIV)

Обучающееся векторное квантование – 1

Векторное квантование



Векторное квантование — метод дискретной аппроксимации непрерывных входных векторов данных $\mathbf{x} \in \mathbb{R}^n$ с помощью конечного числа *кодирующих векторов* $\mathbf{m}_i \in \mathbb{R}^n$, $i = 1, \dots, k$.

Мозаика Вороного — разбивает «пространство образов» на области вокруг *опорных векторов* (точки на рисунке). Все векторы из одной области имеют один и тот же опорный вектор, являющийся их *ближайшим соседом*.

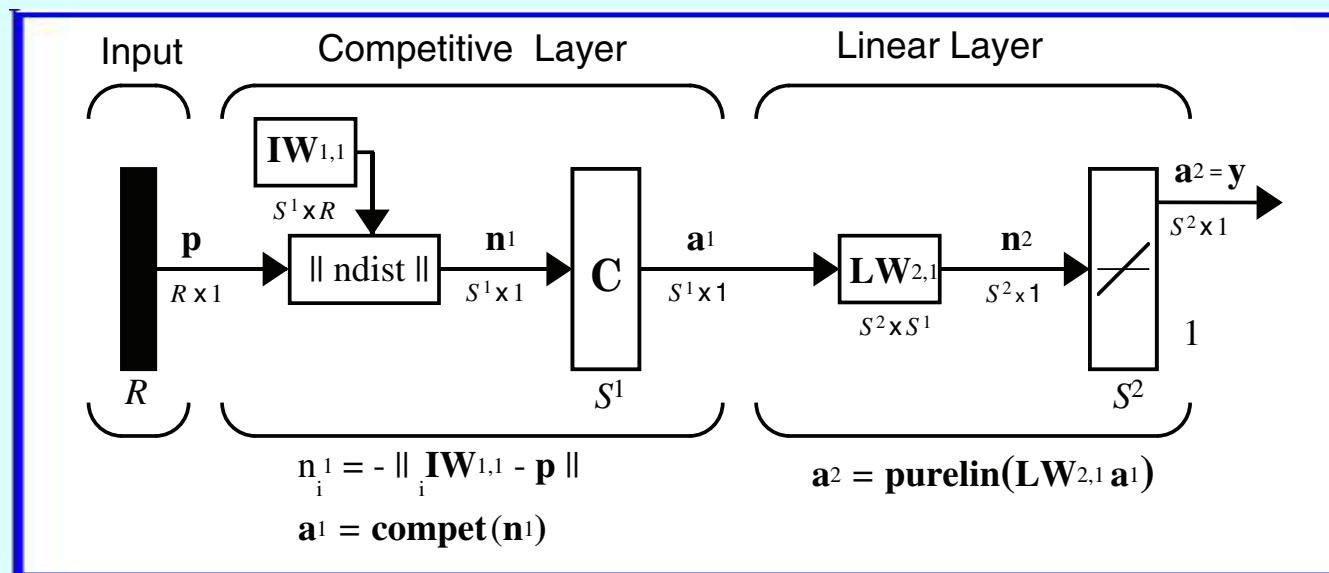
Источник: *Kohonen T.* Self-organizing maps: 3rd Ed. – Springer, 2001. – xx + 501 pp.
(Fig. 1.10, p. 60).

Рус. перевод: *Кохонен Т.* Самоорганизующиеся карты. – М.: БИНОМ. Лаборатория знаний, 2008. – 655 с.

Модели с соревновательным обучением (ХV)

Обучающееся векторное квантование – 2

Сеть LVQ – Learning Vector Quantization



Сеть **LVQ** ориентирована на решение **задач классификации** (распознавания), т.е. она должна разделить пространство входных данных на **области**, отвечающие классам.

Источник: *Demuth H., Beale M., Hagan M.* Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (pp. 9-26, 9-27, 9-28).

Модели с соревновательным обучением (XXVI)

Обучающееся векторное квантование – 3

Сеть LVQ – Learning Vector Quantization

Сеть LVQ в целом:

Делит пространство входных данных **на области**, отвечающие классам.

Соревновательный слой — воспринимает входные векторы и выполняет их **кластеризацию**, выделяя заданное число подклассов **n** .

Линейный слой — объединяет эти подклассы в **m** классов, **$n \geq m$** , в ходе **обучения с учителем**, используя информацию о принадлежности входных векторов определенным классам.

Область, отвечающая определенному классу, **не обязана быть односвязной**.

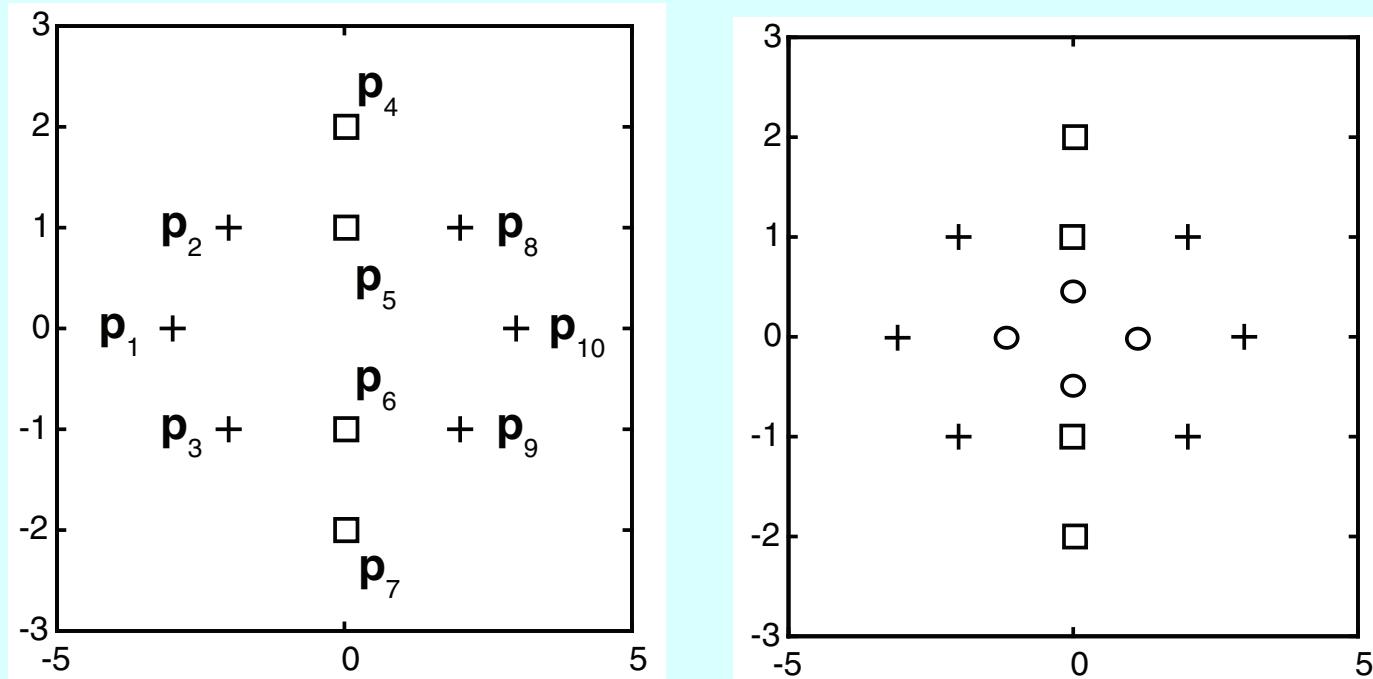
См.: *Kohonen T.* Self-organizing maps: 3rd Ed. – Springer, 2001. – xx + 501 pp.
(Chapter 6, pp. 245–261).

Рус. перевод: *Кохонен Т.* Самоорганизующиеся карты. – М.: БИНОМ. Лаборатория знаний, 2008. – 655 с. (Глава 6, с. 338–361)

Модели с соревновательным обучением (XXVII)

Обучающееся векторное квантование – 4

Сеть LVQ – Learning Vector Quantization



Входные данные – 10 векторов, которые надо разделить на **четыре** подкласса (символ **о** – опорные векторы).

Сеть LVQ – **два** выходных класса.

Источник: Demuth H., Beale M., Hagan M. Neural network toolbox 6: User's guide. – The Mathworks, Inc. – 2009. – 906 pp. (pp. 9-37, 9-41).

Ведущие журналы по нейросетевой тематике:

IEEE Transactions on Neural Networks and Learning Systems

(IEEE Transactions on Neural Networks – по 2011 год)

Institute of Electrical and Electronics Engineers

IEEE Computational Intelligence Society

издается с 1990 года, с 2008 года – 12 номеров в год

Neural Computation

MIT Press

издается с 1989 г., с 2000 года – 12 номеров в год

Neural Networks

The Official Journal of the International Neural Network Society

European Neural Network Society

Japanese Neural Network Society

Elsevier, издается с 1988 г., 10 номеров в год.

Ведущие конференции по нейросетевой тематике:

International Joint Conference on Neural Networks (IJCNN)

IEEE Computational Intelligence Society

International Neural Network Society

ежегодно, с 1987 года

International Conference on Artificial Neural Networks (ICANN)

European Neural Network Society (ENNS)

International Neural Network Society

Japanese Neural Network Society

ежегодно, с 1991 года

Neural Information Processing Systems (NIPS)

Neural Information Processing Systems Foundation

ежегодно, с 1987 года