

Федеральное агентство по образованию  
Государственное образовательное учреждение высшего профессионального образования  
*Амурский государственный университет*  
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ

Зав. кафедрой ИиУС

\_\_\_\_\_ А.В. Бушманов

«\_\_» \_\_\_\_\_ 2007 г.

**Учебно-методический комплекс**

по дисциплине «Нейроинформатика»

для студентов специальностей

230102 – Автоматизированные системы обработки информации и управления;

230201 – Информационные системы и технологии

Составитель: доцент кафедры ИУС Акилова И.М.

Факультет математики и информатики

Кафедра информационных и управляющих систем

2007 г.

*Печатается по решению  
редакционно-издательского совета  
факультета математики и информатики  
Амурского государственного  
университета*

*И.М.Акилова*

Нейроинформатика для специальностей 230102 «Автоматизированные системы обработки информации и управления», 230201 «Информационные системы и технологии»: учебно-методический комплекс дисциплины. / Акилова И.М. – Благовещенск. Изд-во Амурского гос. ун-та, 2007. 158 с.

Пособие содержит рабочую программу, курс лекций, методические рекомендации по проведению и выполнению практических и лабораторных работ. Составлено в соответствии с требованиями государственного образовательного стандарта.

©Амурский государственный университет, 2007

©Кафедра информационных и управляющих систем, 2007

# 1. РАБОЧИЕ ПРОГРАММЫ

## РАБОЧАЯ ПРОГРАММА

По дисциплине «Нейроинформатика»

Для специальности 230102 – «Автоматизированные системы обработки информации и управления»

Курс 4 семестр 8

Лекции 30 (час.) Экзамен -

Практические (семинарские) занятия 30 (час.) Зачет 8 семестр

Лабораторные занятия 15 (час.)

Самостоятельная работа 70 (час.)

Всего часов 145 час.

Составитель доцент Акилова Ирина Михайловна

Факультет математики и информатики

Кафедра Информационных и управляющих систем

# 1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ, ЕЕ МЕСТО В УЧЕБНОМ ПРОЦЕССЕ

## 1.1. Цели и задачи дисциплины

Данная дисциплина имеет своей целью:

- дать представление о прикладных программных средствах, основанных на нейронных сетях;
- дать представления об инструментальном ПО для обучения нейронных сетей и экспериментов с ними;
- подготовить студентов к использованию нейросетевых технологий в научно-исследовательской деятельности.

В результате изучения дисциплины у студентов должны быть сформированы представления о:

- разных моделях нейронных сетей, их особенностях;
- особенностях решения задач с помощью нейронных сетей;
- существующих прикладных системах, основанных на применении нейронных сетей;
- этапах решения задач с помощью нейронных сетей.

После изучения дисциплины студент должен знать содержание:

- принципов постановки задач для решения с помощью нейронных сетей;
- методов представления данных для обучения и использования нейронных сетей;
- методов обучения нейронных сетей и оценки качества обучения нейронной сети;
- этапов решения задач с помощью нейронных сетей.

После изучения дисциплины студент должен приобрести умения и навыки:

- ориентироваться в различных типах прикладных систем, основанных на использовании нейронных сетей;
- ориентироваться а различных методах представления данных для обучения нейронной сети;
- выбирать и ставить задачу для решения ее нейронной сетью;

- выбирать модель нейронной сети для решения задачи.

## 2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

### 2.1. Федеральный компонент

Программа курса «Нейроинформатика» составлена в соответствии с требованиями государственного образовательного стандарта специализации – Автоматизированные системы обработки информации и управления, специализации 230102, блок дисциплины по выбору ЕН.В.01.

### 2.2. Наименование тем, их содержание, объем в лекционных часах

#### ТЕМАТИЧЕСКИЙ ПЛАН ЛЕКЦИОННЫХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Введение	4
2	Модели нейронов и методы их обучения	2
3	Однонаправленные многослойные сети сигмоидального типа	2
4	Проблемы практического использования искусственных нейронных сетей и их свойства	2
5	Радиальные нейронные сети	4
6	Рекуррентные сети как ассоциативные запоминающие устройства	4
7	Рекуррентные сети на базе перцептрона	4
8	Сети с самоорганизацией на основе конкуренции	4
9	Сеть с самоорганизацией корреляционного типа	2
10	Вероятностная нейронная сеть	2
Итого		30

#### 1. ВВЕДЕНИЕ – 2 часа.

Биологические основы функционирования нейрона. Первые модели нейронной сети. Прикладные возможности нейронных сетей. Определение искусственных нейронных сетей. Свойства биологических и искусственных нейронных сетей. Способы реализации нейросетей. Типы задач, решаемых нейронными сетями. Недостатки и ограничения нейронных сетей.

#### 2. МОДЕЛИ НЕЙРОНОВ И МЕТОДЫ ИХ ОБУЧЕНИЯ. – 1 час.

Перцептрон. Сигмоидальный нейрон. Нейрон типа «адалайн». Инстар и оутстар Гроссберга. Нейроны типа WTA. Модель нейрона Хебба. Стахостическая модель нейрона.

### 3. ОДНОНАПРАВЛЕННЫЕ МНОГОСЛОЙНЫЕ СЕТИ СИГМОИДАЛЬНОГО ТИПА. – 1 час.

Однослойная сеть. Многослойный перцептрон. Структура перцептронной сети. Алгоритм обратного распространения ошибки. Градиентные алгоритмы обучения сети: основные положения, алгоритм наискорейшего спуска, алгоритм переменной метрики, алгоритм Левенберга-Марквардта, алгоритм сопряженных градиентов. Подбор коэффициента обучения. Методы инициализации весов.

### 4. ПРОБЛЕМЫ ПРАКТИЧЕСКОГО ИСПОЛЬЗОВАНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ И ИХ СВОЙСТВА . – 1 час.

Предварительный подбор архитектуры сети. Подбор оптимальной архитектуры сети. Методы наращивания сети. Подбор обучающих выборок. Добавление шума в обучающие выборки. Распознавание и классификация образов. Нейронная сеть для сжатия данных. Идентификация динамических объектов.

### 5. РАДИАЛЬНЫЕ НЕЙРОННЫЕ СЕТИ. – 2 часа.

Математические основы. Радиальная нейронная сеть. Методы обучения радиальных нейронных сетей. Пример использования радиальной сети. Методы подбора количества базисных функций: эвристические методы, метод ортогонализации Грэма-Шмидта.

### 6. РЕКУРРЕНТНЫЕ СЕТИ КАК АССОЦИАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА. – 2 часа.

Сети с обратными связями. Послойность сети и матричное умножение. Расчет градиента квадратичной формы. Выбор начальной точки и длины шага. Сеть Хопфилда. Сеть Хемминга. Сеть АРТ –1

### 7. РЕКУРРЕНТНЫЕ СЕТИ НА БАЗЕ ПЕРЦЕПТРОНА – 2 часа.

Перцептронная сеть с обратной связью: структура сети RMLP, алгоритм обучения сети RMLP, подбор коэффициента обучения, коэффициент усиления

сигнала. Рекуррентная сеть Элмана: структура сети, алгоритм обучения сети Элмана, обучение с учетом момента.

8. СЕТИ С САМООРГАНИЗАЦИЕЙ НА ОСНОВЕ КОНКУРЕНЦИИ . – 2 часа.

Нейронные сети встречного распространения. Сети Кохонена. Обучение слоя Кохонена. Примеры обучения сети Кохонена. Применение сети Кохонена для сжатия данных. Слой Гроссбера: обучение слоя Гроссбера, модификации.

9. СЕТЬ С САМООРГАНИЗАЦИЕЙ КОРРЕЛЯЦИОННОГО ТИПА. – 1 час.

Энергетическая функция корреляционных сетей. Нейронные сети РСА: математическое введение, определение первого главного элемента, алгоритм определения множества главных компонентов.

10. ВЕРОЯТНОСТНАЯ НЕЙРОННАЯ СЕТЬ. – 1 час.

Сети, использующие статистический подход. Метод «модельной закалки». Пример алгоритма минимизации функции. Машина Больцмана. Архитектура нейронной сети PNN. Пример модульной нейронной сети.

### 2.3. Практические занятия, их содержание и объем в часах.

#### ТЕМАТИЧЕСКИЙ ПЛАН ПРАКТИЧЕСКИХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Структура связей. Весовые матрицы и их составление. Правило вычисления сигнала активности. Отношение XOR.	6
2	Примеры обучения сети с использованием алгоритма обратного распространения ошибки	4
3	Однонаправленные многослойные сети сигмоидального типа	6
4	Сети с самоорганизацией на основе конкуренции. Нормализация обучающих векторов. Алгоритм Кохонена.	4
5	Описание проблемы классификации и нейронных сетей, способных выполнять классификацию образов	6
6	Ассоциация образцов. Описание ассоциативной памяти. Автоассоциативная сеть Хопфилда.	4

	Всего	30
--	-------	----

#### 2.4. Лабораторные занятия, их наименование и объем в часах.

##### ТЕМАТИЧЕСКИЙ ПЛАН ЛАБОРАТОРНЫХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Изучение пакета расширения по нейронным сетям системы MATLAB.	2
2	Создание однонаправленной сети.	2
3	Многослойный перцептрон.	4
4	Сеть Кохонена	2
5	Сеть Хопфилда	3
6	Сеть Элмана	2
	Всего	15

#### 2.4. Самостоятельная работа студентов (70 часов)

##### ТЕМЫ САМОСТОЯТЕЛЬНЫХ РАБОТ

1. Предсказание стационарного сигнала
2. Слежение за нестационарным сигналом
3. Моделирование стационарного фильтра
4. Моделирование нестационарного фильтра
5. Распознавание образов
6. Регулятор с предсказанием
7. Регулятор NARMA-L2
8. Регулятор на основе эталонной модели

#### 2.5. ВОПРОСЫ К ЗАЧЕТУ

1.Что такое нейронные сети (НС)? Что дает моделирование НС? Проблемы, возникающие при моделировании. Свойства биологических и искусственных НС. Способы реализации нейросетей.

2.Место НС среди других методов решения задач. Типы задач, решаемых нейронными сетями. Недостатки и ограничения НС.

3.Биологический нейрон. Структура, функции.



- 4.Формальный нейрон. Виды функций активации. Ограниченность модели формального нейрона.
  - 5.Многослойный перцептрон. Структура, алгоритм работы. Этапы решения задачи с помощью НС.
  - 6.Формализация условий задачи для НС. Примеры. Подготовка входных и выходных данных. Выбор количества слоев.
  - 7.Обучение однослойного перцептрона. Выбор шагов по  $W, \Theta$ .
  - 8.Проблема "исключающего ИЛИ" и ее решение.
  - 9.Перцептронная представляемость.
  - 10.Метод обратного распространения ошибки.
  - 11.Паралич сети. Выбор шага по параметрам. Локальные минимумы. Временная неустойчивость.
  - 12.Примеры применения перцептронов.
  - 13.Динамическое добавление нейронов. Способность НС к обобщению.
  - 14.Обучение без учителя. Сеть с линейным поощрением.
  - 15.Задача классификации. Сеть Кохонена.
  - 16.Обучение слоя Кохонена. Метод выпуклой комбинации. Примеры обучения.
  - 17.Режимы работы сети Кохонена. Применение для сжатия данных.
  - 18.Сеть встречного распространения. Схема, обучение, свойства.
  - 19.Генетические алгоритмы для обучения НС. Положительные качества и недостатки.
  - 20.Послойность сети и матричное умножение. Расчет градиента квадратичной формы с помощью НС. Выбор начальной точки и длины шага.
  - 21.Сети с обратными связями. Сеть Хопфилда. Вычислительная энергия и ее минимизация.
  - 22.Этапы решения задачи сетью Хопфилда. Устойчивость, сходимость к эталонам.
  - 23.Соотношение стабильности пластичности при запоминании. Сеть ART
1. Структура, описание элементов сети.

24.Работа сети АРТ 1. Запоминание и классификация векторов сетью.

25.Метод имитации отжига.

### 2.7. Виды контроля

Для проверки эффективности преподавания дисциплины проводится контроль знаний студентов.

При этом используются следующие виды контроля:

- текущий контроль за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения аудиторных занятий посредством устного опроса;

- промежуточный контроль осуществляется два раза в семестр в виде анализа итоговых отчетов;
- итоговый контроль в виде зачета осуществляется после успешного прохождения студентами текущего и промежуточного контроля и сдачи отчета по самостоятельной работе и устного экзамена при ответах экзаменуемого на два вопроса в билете и дополнительные вопросы по желанию экзаменатора.

### 2.8. Требования к знаниям студентов, предъявляемые на зачете

Для получения зачета студент должен посещать занятия, студентом должны быть выполнены все лабораторные работы, самостоятельное задание и отчет к нему, а также знание теоретического материала в объеме лабораторного курса.

«Зачтено» - студент не имеет задолженностей по семестровым отчетным работам. Хорошо владеет теоретическим материалом.

«Незачтено» - студент не отчитался по семестровым отчетным работам, не знает теоретический материал.

## 3. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

**Основная:**

1. Оссовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2002.
2. Медведев В.С., Потемкин В.Г. Нейронные сети. Matlab 6. М.: Диалог МИФИ, 2002.
3. Роберт Каллан. Основные концепции нейронных сетей. М.: Издательский дом «Вильямс», 2001.
4. Суровцев И.С., Ключкин В.И., Пивоварова Р.П. Нейронные сети. - Воронеж: ВГУ, 1994.
5. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика. М.: Мир, 1992.

**Дополнительная:**

1. Горбань А.Н. и др. Нейроинформатика.- Электронная публикация.
2. Мкртчян С.О. Нейроны и нейронные сети. (Введение в теорию формальных нейронов)— М.: Энергия, 1971.
3. А.Н. Горбань, Д.А. Россиев. Нейронные сети на персональном компьютере. -Новосибирск, Наука, 1996.
4. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. М.: Мир, 1985.
5. Лоскутов А.Ю., Михайлов А.С. Введение в синергетику. М.: Наука. Гл. ред. физ. мат. лит., 1990.— 272 с.
6. КуССуль Э.М. Ассоциативные нейроподобные структуры. - Киев, Наукова думка, 1990.
7. Нейрокомпьютеры и интеллектуальные роботы. Под ред. Н.М.Амосова. – Киев, Наукова думка, 1991.

#### **4. НЕОБХОДИМОЕ ТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**

Лекции проводятся в стандартной аудитории, оснащенной в соответствии с требованиями преподавания теоретических дисциплин.

Для проведения лабораторных работ необходим компьютерный класс на 12-14 посадочных мест пользователей. В классе должен быть установлен пакет Matlab 6.12.

### 5. Учебно-методическая (технологическая) карта дисциплины

Номер недели	Номер темы	изучаемые на лекции Вопросы,	Занятия		Используемые наглядные и методические пособия	Самостоятельная работа студентов		Форма контроля
			Практические	Лабораторные		Содержание	Часы	
1	2	3	4	5	6	7	8	9
1	1	1-4	1	1	1,2-осн.	Изучение литературы по нейроинформатике	35	
2		5-8	1	1	1,2-осн.			злр.
3	2	1-8	1	2	1,2-осн.			
4	3	1-7	2	2	1,2-осн.			злр.
5	4	1-8	2	3	1,2-осн.			
6	5	1-3	3	3	1,2-осн.			
7		4-5	3	3	1,2-осн.			злр.
8	6	1-3	3	3	1,2-осн.	Выполнение задания для самостоятельной работы	31	злр.
9		4-7	4	4	1,2-осн.			
10	7	1	4	4	1,2-осн.			злр.
11		2	5	5	1,2-осн.			
12	8	1-3	5	5	1,2-осн.			
13		4-7	5	5	1,2-осн.			злр.
14	9	1-2	6	6	1,2-осн.			
15	10	1-6	6	6	1,2-осн.	сб.	4	злр.

Условные обозначения:

осн. – основная литература

злр - защита лабораторной работы

сб. - собеседование по результатам самостоятельной работы студентов

## РАБОЧАЯ ПРОГРАММА

По дисциплине «Нейроинформатика»

Для специальности 230201 – «Информационные системы и технологии»

Курс 3 семестр 5

Лекции 36 (час.) Экзамен – 5 семестр

Практические (семинарские) занятия - (час.) Зачет -

Лабораторные занятия 18 (час.)

Самостоятельная работа 66 (час.)

Всего часов 120 час.

Составитель: доцент Акилова И.М.

Факультет математики и информатики

Кафедра Информационных и управляющих систем

2006

## 1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ, ЕЕ МЕСТО В УЧЕБНОМ ПРОЦЕССЕ

### 1.1. Цели и задачи дисциплины

Данная дисциплина имеет своей целью:

- дать представление о прикладных программных средствах, основанных на нейронных сетях;
- дать представления об инструментальном ПО для обучения нейронных сетей и экспериментов с ними;
- подготовить студентов к использованию нейросетевых технологий в научно-исследовательской деятельности.

В результате изучения дисциплины у студентов должны быть сформированы представления о:

- разных моделях нейронных сетей, их особенностях;
- особенностях решения задач с помощью нейронных сетей;
- существующих прикладных системах, основанных на применении нейронных сетей;
- этапах решения задач с помощью нейронных сетей.

После изучения дисциплины студент должен знать содержание:

- принципов постановки задач для решения с помощью нейронных сетей;
- методов представления данных для обучения и использования нейронных сетей;

- методов обучения нейронных сетей и оценки качества обучения нейронной сети;

- этапов решения задач с помощью нейронных сетей.

После изучения дисциплины студент должен приобрести умения и навыки:

- ориентироваться в различных типах прикладных систем, основанных на использовании нейронных сетей;

- ориентироваться в различных методах представления данных для обучения нейронной сети;

- выбирать и ставить задачу для решения ее нейронной сетью;

- выбирать модель нейронной сети для решения задачи.

## 2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

### 2.1. Федеральный компонент

Программа курса «Нейроинформатика» составлена в соответствии с требованиями государственного образовательного стандарта специализации – Информационные системы и технологии, специализации 230201, блок дисциплины по выбору ЕН.В.01.

### 2.2. Наименование тем, их содержание, объем в лекционных часах

#### ТЕМАТИЧЕСКИЙ ПЛАН ЛЕКЦИОННЫХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Введение	6
2	Модели нейронов и методы их обучения	4
3	Однонаправленные многослойные сети сигмоидального типа	2
4	Проблемы практического использования искусственных нейронных сетей и их свойства	2
5	Радиальные нейронные сети	4
6	Рекуррентные сети как ассоциативные запоминающие устройства	4
7	Рекуррентные сети на базе перцептрона	4

8	Сети с самоорганизацией на основе конкуренции	6
9	Сеть с самоорганизацией корреляционного типа	2
10	Вероятностная нейронная сеть	2
Итого		36

### 1. ВВЕДЕНИЕ – 6 часов.

Биологические основы функционирования нейрона. Первые модели нейронной сети. Прикладные возможности нейронных сетей. Определение искусственных нейронных сетей. Свойства биологических и искусственных нейронных сетей. Способы реализации нейросетей. Типы задач, решаемых нейронными сетями. Недостатки и ограничения нейронных сетей.

### 2. МОДЕЛИ НЕЙРОНОВ И МЕТОДЫ ИХ ОБУЧЕНИЯ. – 4 часа.

Перцептрон. Сигмоидальный нейрон. Нейрон типа «адалайн». Инстар и оутстар Гроссберга. Нейроны типа WTA. Модель нейрона Хебба. Стахостическая модель нейрона.

### 3. ОДНОНАПРАВЛЕННЫЕ МНОГОСЛОЙНЫЕ СЕТИ СИГМОИДАЛЬНОГО ТИПА. – 2 часа.

Однослойная сеть. Многослойный перцептрон. Структура перцептронной сети. Алгоритм обратного распространения ошибки. Градиентные алгоритмы обучения сети: основные положения, алгоритм наискорейшего спуска, алгоритм переменной метрики, алгоритм Левенберга-Марквардта, алгоритм сопряженных градиентов. Подбор коэффициента обучения. Методы инициализации весов.

### 4. ПРОБЛЕМЫ ПРАКТИЧЕСКОГО ИСПОЛЬЗОВАНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ И ИХ СВОЙСТВА . – 2 часа.

Предварительный подбор архитектуры сети. Подбор оптимальной архитектуры сети. Методы наращивания сети. Подбор обучающих выборок. Добавление шума в обучающие выборки. Распознавание и классификация образов. Нейронная сеть для сжатия данных. Идентификация динамических объектов.

### 5. РАДИАЛЬНЫЕ НЕЙРОННЫЕ СЕТИ. – 4 часа.

Математические основы. Радиальная нейронная сеть. Методы обучения радиальных нейронных сетей. Пример использования радиальной сети. Методы



подбора количества базисных функций: эвристические методы, метод ортогонализации Грэма-Шмидта.

#### 6. РЕКУРРЕНТНЫЕ СЕТИ КАК АССОЦИАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА. – 4 часа.

Сети с обратными связями. Послойность сети и матричное умножение. Расчет градиента квадратичной формы. Выбор начальной точки и длины шага. Сеть Хопфилда. Сеть Хемминга. Сеть АРТ –1

#### 11. РЕКУРРЕНТНЫЕ СЕТИ НА БАЗЕ ПЕРЦЕПТРОНА – 4 часа.

Перцептронная сеть с обратной связью: структура сети RMLP, алгоритм обучения сети RMLP, подбор коэффициента обучения, коэффициент усиления сигнала. Рекуррентная сеть Элмана: структура сети, алгоритм обучения сети Элмана, обучение с учетом момента.

#### 7. СЕТИ С САМООРГАНИЗАЦИЕЙ НА ОСНОВЕ КОНКУРЕНЦИИ . – 6 часов.

Нейронные сети встречного распространения. Сети Кохонена. Обучение слоя Кохонена. Примеры обучения сети Кохонена. Применение сети Кохонена для сжатия данных. Слой Гроссбера: обучение слоя Гроссбера, модификации.

#### 8. СЕТЬ С САМООРГАНИЗАЦИЕЙ КОРРЕЛЯЦИОННОГО ТИПА. – 2 часа.

Энергетическая функция корреляционных сетей. Нейронные сети РСА: математическое введение, определение первого главного элемента, алгоритм определения множества главных компонентов.

#### 9. ВЕРОЯТНОСТНАЯ НЕЙРОННАЯ СЕТЬ. – 2 часа.

Сети, использующие статистический подход. Метод «модельной закалки». Пример алгоритма минимизации функции. Машина Больцмана. Архитектура нейронной сети PNN. Пример модульной нейронной сети.

2.4. Лабораторные занятия, их наименование и объем в часах.

## ТЕМАТИЧЕСКИЙ ПЛАН ЛАБОРАТОРНЫХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Изучение пакета расширения по нейронным сетям системы MATLAB.	2
2	Создание однонаправленной сети.	2
3	Многослойный перцептрон.	5
4	Сеть Кохонена	4
5	Сеть Хопфилда	3
6	Сеть Элмана	2
	Всего	18

#### 2.4. Самостоятельная работа студентов (66 часов)

##### ТЕМЫ САМОСТОЯТЕЛЬНЫХ РАБОТ

1. Предсказание стационарного сигнала
2. Слежение за нестационарным сигналом
3. Моделирование стационарного фильтра
4. Моделирование нестационарного фильтра
5. Распознавание образов
6. Регулятор с предсказанием
7. Регулятор NARMA-L2
1. Регулятор на основе эталонной модели

#### 2.5. ВОПРОСЫ К ЭКЗАМЕНУ

1. Что такое нейронные сети (НС)? Что дает моделирование НС? Проблемы, возникающие при моделировании. Свойства биологических и искусственных НС. Способы реализации нейросетей.

2. Место НС среди других методов решения задач. Типы задач, решаемых нейронными сетями. Недостатки и ограничения НС.

3. Биологический нейрон. Структура, функции.

4. Формальный нейрон. Виды функций активации. Ограниченность модели формального нейрона.

5. Многослойный перцептрон. Структура, алгоритм работы. Этапы решения задачи с помощью НС.
6. Формализация условий задачи для НС. Примеры. Подготовка входных и выходных данных. Выбор количества слоев.
7. Обучение однослойного перцептрона. Выбор шагов по  $W, \Theta$ .
8. Проблема "исключающего ИЛИ" и ее решение.
9. Перцептронная представляемость.
10. Метод обратного распространения ошибки.
11. Паралич сети. Выбор шага по параметрам. Локальные минимумы. Временная неустойчивость.
12. Примеры применения перцептронов.
13. Динамическое добавление нейронов. Способность НС к обобщению.
14. Обучение без учителя. Сеть с линейным поощрением.
15. Задача классификации. Сеть Кохонена.
16. Обучение слоя Кохонена. Метод выпуклой комбинации. Примеры обучения.
17. Режимы работы сети Кохонена. Применение для сжатия данных.
18. Сеть встречного распространения. Схема, обучение, свойства.
19. Генетические алгоритмы для обучения НС. Положительные качества и недостатки.
20. Послойность сети и матричное умножение. Расчет градиента квадратичной формы с помощью НС. Выбор начальной точки и длины шага.
21. Сети с обратными связями. Сеть Хопфилда. Вычислительная энергия и ее минимизация.
22. Этапы решения задачи сетью Хопфилда. Устойчивость, сходимости к эталонам.
23. Соотношение стабильности пластичности при запоминании. Сеть ART 1.
  1. Структура, описание элементов сети.
24. Работа сети ART 1. Запоминание и классификация векторов сетью.
25. Метод имитации отжига.

## 2.7. Виды контроля

Для проверки эффективности преподавания дисциплины проводится контроль знаний студентов.

При этом используются следующие виды контроля:

- текущий контроль за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения аудиторных занятий посредством устного опроса;
- промежуточный контроль осуществляется два раза в семестр в виде анализа итоговых отчетов;
- итоговый контроль в виде зачета осуществляется после успешного прохождения студентами текущего и промежуточного контроля и сдачи отчета по самостоятельной работе и устного экзамена при ответах экзаменуемого на два вопроса в билете и дополнительные вопросы по желанию экзаменатора.

## 2.9. Требования к знаниям студентов, предъявляемые на экзамене

Знания студента оцениваются как отличные при полном изложении теоретического материала экзаменационного билета и ответах на дополнительные вопросы со свободной ориентацией в материале и других литературных источниках.

Оценка «хорошо» ставится при твердых знаниях студентом всех разделов курса, но в пределах конспекта лекций и обязательных заданий по самостоятельной работе с литературой.

Оценку «удовлетворительно» студент получает, если даны неполные ответы на теоретические вопросы билета, показывая поверхностные знания учебного материала, владение основными понятиями и терминологией; при неверном ответе на билет ответы на наводящие вопросы.

Оценка «неудовлетворительно» выставляется за незнание студентом одного из разделов курса. Студент не дает полные ответы на теоретические во-

просы билета, показывая лишь фрагментарное знание учебного материала, незнание основных понятий и терминологии; наводящие вопросы остаются без ответа.

### 3. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная:

1. Оссовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2002.
2. Медведев В.С., Потемкин В.Г. Нейронные сети. Matlab 6. М.: Диалог МИФИ, 2002.
3. Роберт Каллан. Основные концепции нейронных сетей. М.: Издательский дом «Вильямс», 2001.
4. Суровцев И.С., Ключкин В.И., Пивоварова Р.П. Нейронные сети. - Воронеж: ВГУ, 1994.
5. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика. М.: Мир, 1992.

Дополнительная:

1. Горбань А.Н. и др. Нейроинформатика.- Электронная публикация.
2. Мкртчян С.О. Нейроны и нейронные сети. (Введение в теорию формальных нейронов)— М.: Энергия, 1971.
3. А.Н. Горбань, Д.А. Россиев. Нейронные сети на персональном компьютере. -Новосибирск, Наука, 1996.
4. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. М.: Мир, 1985.
5. Лоскутов А.Ю., Михайлов А.С. Введение в синергетику. М.: Наука. Гл. ред. физ. мат. лит., 1990.— 272 с.
6. КуССуль Э.М. Ассоциативные нейроподобные структуры. - Киев, Наукова думка, 1990.
7. Нейрокомпьютеры и интеллектуальные роботы. Под ред. Н.М.Амосова. – Киев, Наукова думка, 1991.

#### 4. НЕОБХОДИМОЕ ТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Лекции проводятся в стандартной аудитории, оснащенной в соответствии с требованиями преподавания теоретических дисциплин.

Для проведения лабораторных работ необходим компьютерный класс на 12-14 посадочных мест пользователей. В классе должен быть установлен пакет Matlab 6.12.

#### 5. Учебно-методическая (технологическая) карта дисциплины

Номер недели	Номер темы	изучаемые на лекции Вопросы,	Занятия		Используемые наглядные и методические пособия	Самостоятельная работа студентов		Форма контроля
			Практические	Лабораторные		Содержание	Часы	
1	2	3	4	5	6	7	8	9
1	1	1-2	-	1	1,2-осн.	Изучение литературы по нейроинформатике	30	
2	1	3-5	-	1	1,2-осн.			злр.
3	1	6-8	-	2	1,2-осн.			
4	2	1-4	-	2	1,2-осн.			злр.
5	2	5-7	-	3	1,2-осн.			
6	3	1-7	-	3	1,2-осн.			злр.
7	4	1-7	-	3	1,2-осн.			

8	5	1-3	-	3	1,2-осн.	Выполнение задания для самостоятельной работы	32	
9	5	4-5	-	3	1,2-осн.			злр.
10	6	1-3	-	4	1,2-осн.			
11	6	4-6	-	4	1,2-осн.			
12	7	1	-	4	1,2-осн.			
13	7	2	-	4	1,2-осн.			злр.
14	8	1-2	-	5	1,2-осн.			
15	8	3-5	-	5	1,2-осн.			
16	8	6	-	5	1,2-осн.			злр.
17	9	1-2	-	6	1,2-осн.			
18	10	1-6	-	6	1,2-осн.	сб.	4	злр.

Условные обозначения:

осн. – основная литература

злр - защита лабораторной работы

сб. - собеседование по результатам самостоятельной работы студентов

### 3. ГРАФИК САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Содержание	Объем в часах	Сроки и форма контроля
Предсказание стационарного сигнала	10/9 час.	Собеседование
Слежение за нестационарным сигналом	10/9 час.	Собеседование
Моделирование стационарного фильтра	10/8 час.	Собеседование
Моделирование нестационарного фильтра	8/8 час.	Собеседование
Распознавание образов	8/8 час.	Собеседование
Регулятор с предсказанием	8/8 час.	Собеседование
Регулятор NARMA-L2	8/8 час.	Собеседование
Регулятор на основе эталонной модели	8/8 час.	Собеседование

#### **4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ**

По выбранной теме студенты выполняют реферативную работу.

Реферативная работа включает следующие разделы:

1. Обоснование актуальности выбранной тематики и описание целей выполнения работы.
2. Систематизация и анализ найденных в научной печати, в сети Интернет и других источниках материалов.
3. Выводы.
4. Предложения по использованию результатов работы в конкретных областях и возможные направления дальнейших исследований.



## **5. ПЕРЕЧЕНЬ УЧЕБНИКОВ, УЧЕБНЫХ ПОСОБИЙ**

### **5.1. Перечень основной литературы**

1. Оссовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2002.
2. Медведев В.С., Потемкин В.Г. Нейронные сети. Matlab 6. М.: Диалог МИФИ, 2002.
3. Роберт Каллан. Основные концепции нейронных сетей. М.: Издательский дом «Вильямс», 2001.
4. Суровцев И.С., Ключкин В.И., Пивоварова Р.П. Нейронные сети. - Воронеж: ВГУ, 1994.
5. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика. М.: Мир, 1992.

### **5.2. Перечень дополнительной литературы**

1. Горбань А.Н. и др. Нейроинформатика.- Электронная публикация.
2. Мкртчян С.О. Нейроны и нейронные сети. (Введение в теорию формальных нейронов)— М.: Энергия, 1971.
3. А.Н. Горбань, Д.А. Россиев. Нейронные сети на персональном компьютере. -Новосибирск, Наука, 1996.
4. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. М.: Мир, 1985.
5. Лоскутов А.Ю., Михайлов А.С. Введение в синергетику. М.: Наука. Гл. ред. физ. мат. лит., 1990.— 272 с.
6. КуССуль Э.М. Ассоциативные нейроподобные структуры. - Киев, Наукова думка, 1990.
7. Нейрокомпьютеры и интеллектуальные роботы. Под ред. Н.М.Амосова. – Киев, Наукова думка, 1991.

## 6. КРАТКИЙ КОНСПЕКТ ЛЕКЦИЙ

### Лекция № 1 *Тема: Введение*

Нейронные сети - это раздел искусственного интеллекта, в котором для обработки сигналов используются явления, аналогичные происходящим в нейронах живых существ.

#### **Свойства нейронных сетей**

Важнейшее свойство нейронных сетей, свидетельствующее об их огромном потенциале и широких прикладных возможностях, состоит в параллельной обработке информации одновременно всеми нейронами. Благодаря этой способности при большом количестве межнейронных связей достигается значительное ускорение процесса обработки информации. Во многих ситуациях становится возможной обработка сигналов в реальном масштабе времени.

Кроме того, при большом числе межнейронных соединений сеть приобретает устойчивость к ошибкам, возникающим на некоторых линиях. Функции поврежденных связей берут на себя исправные линии, в результате чего деятельность сети не претерпевает существенных возмущений. Это свойство используется, в частности, при поиске оптимальной архитектуры нейронной сети путем разрыва отдельных связей.

Другое не менее важное свойство - способность к обучению и обобщению накопленных знаний. Нейронная сеть обладает чертами искусственного интеллекта. Натренированная на ограниченном множестве данных сеть способна обобщать полученную информацию и показывать хорошие результаты на данных, не использовавшихся в процессе обучения.

Характерная особенность сети состоит также в возможности ее реализации с применением технологии сверхбольшой степени интеграции. Различие элементов сети невелико, а их повторяемость огромна. Это открывает перспективу создания универсального процессора с однородной структурой, способного перерабатывать разнообразную информацию.

Использование перечисленных свойств на фоне развития устройств со сверхбольшой степенью интеграции (VLSI) и повсеместного применения вычис-

лительной техники вызвало в последние годы огромный рост интереса к нейронным сетям и существенный прогресс в их исследовании. Создана база для выработки новых технологических решений, касающихся восприятия, искусственного распознавания и обобщения видеоинформации, управления сложными системами, обработки речевых сигналов и т.п. Искусственные нейронные сети в практических приложениях, как правило, используются в качестве подсистемы управления или выработки решений, передающей исполнительный сигнал другим подсистемам, имеющим иную методологическую основу.

### **Прикладные возможности нейронных сетей**

Любая нейронная сеть используется в качестве самостоятельной системы представления знаний, которая в практических приложениях выступает, как правило, в качестве одного из компонентов системы управления либо модуля принятия решений, передающих результирующий сигнал на другие элементы, не связанные непосредственно с искусственной нейронной сетью.

Функции, выполняемые сетями, подразделяются на несколько групп:

1) аппроксимация и интерполяция;

Аппроксимирующая сеть играет роль универсального аппроксиматора функции нескольких переменных, который реализует нелинейную функцию вида  $y = f(x)$ , где  $x$  - входной вектор, а  $y$  - реализованная функция нескольких переменных. Множество задач моделирования, идентификации, обработки сигналов удастся сформулировать в аппроксимационной постановке.

2) классификация и распознавание образов;

Для классификации и распознавания образов сеть накапливает в процессе обучения знания об основных свойствах этих образов, таких, как геометрическое отображение структуры изображения, относительное расположение важнейших элементов образа, компоненты преобразования Фурье и другие подобные факторы. В процессе обучения выделяются признаки, отличающие образы друг от друга, которые и составляют базу для принятия решений об отнесении образов к соответствующим классам.

3) прогнозирование;

В области прогнозирования задача сети формулируется как предсказание будущего поведения системы по имеющейся последовательности ее предыдущих состояний. По информации о значениях переменной  $x$  в моменты времени, предшествующие прогнозированию, сеть вырабатывает решение о том, чему должно быть равно оцениваемое значение исследуемой последовательности в текущий момент времени.

#### 4) идентификация и оценивание;

В задачах управления динамическими процессами нейронная сеть выполняет, как правило, несколько функций. Во-первых, она представляет собой нелинейную модель этого процесса и идентифицирует его основные параметры, необходимые для выработки соответствующего управляющего сигнала. Во-вторых, сеть выполняет функции следящей системы, отслеживает изменяющиеся условия окружающей среды и адаптируется к ним. Она также может играть роль нейрорегулятора, заменяющего собой традиционные устройства. Важное значение, особенно при управлении роботами, имеют классификация текущего состояния и выработка решений о дальнейшем развитии процесса.

#### 5) ассоциативное управление.

В задачах ассоциации нейронная сеть выступает в роли ассоциативного запоминающего устройства. Здесь можно выделить память автоассоциативного типа, в которой взаимосвязности охватывают только конкретные компоненты входного вектора, и память гетероассоциативного типа, с помощью которой сеть определяет взаимосвязи различных векторов. Даже если на вход сети подается вектор, искаженный шумом либо лишенный отдельных фрагментов данных, то сеть способна восстановить полный и очищенный от шумов исходный вектор путем генерации соответствующего ему выходного вектора.

### **Топология нейронных сетей**

Различные способы объединения нейронов между собой и организации их взаимодействия привели к созданию сетей разных типов.

Классифицируя нейронные сети по топологии, можно выделить три основных типа таких сетей:

- полносвязные сети ;
- многослойные, или слоистые сети ;
- слабосвязные сети (нейронные сети с локальными связями).

*Полносвязные сети* представляют собой искусственные нейронные сети, каждый нейрон которой передает свой выходной сигнал остальным нейронам, в том числе и самому себе. *Все* входные сигналы подаются *всем* нейронам. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.

В *многослойных сетях* нейроны объединяются в *слои*. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. В общем случае сеть состоит из  $Q$  слоев, пронумерованных слева направо. Внешние входные сигналы подаются на входы нейронов первого слоя (входной слой часто нумеруют как нулевой), а выходами сети являются выходные сигналы последнего слоя. Вход нейронной сети можно рассматривать как выход «нулевого слоя» вырожденных нейронов, которые служат лишь в качестве распределительных точек, суммирования и преобразования сигналов здесь не производится. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько промежуточных (скрытых) слоев. Связи от выходов нейронов некоторого слоя  $q$  ко входам нейронов следующего слоя ( $q+1$ ) называются последовательными.

В свою очередь среди слоистых сетей выделяют следующие типы:

1. *Монотонные*. Это специальный частный случай слоистых сетей с дополнительными условиями на связи и элементы. Каждый, слой, кроме последнего (выходного), разбит на два блока: возбуждающий (В) и тормозящий (Т). Связи между блоками тоже разделяются на тормозящие и возбуждающие. Если от блока  $A$  к блоку  $C$  ведут только возбуждающие связи, это означает, что любой выходной сигнал блока  $C$  является монотонной неубывающей

функцией любого выходного сигнала блока  $A$ . Если же эти связи только тормозящие, то любой выходной сигнал блока  $C$  является не-возрастающей функцией любого выходного сигнала блока  $A$ . Для элементов монотонных сетей необходима монотонная зависимость выходного сигнала элемента от параметров входных сигналов.

2. *Сети без обратных связей.* В таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам 1-го скрытого слоя, далее срабатывает 1-й скрытый слой и т. д. до  $q$ -го слоя, который выдает выходные сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал  $q$ -го слоя подается на вход всех нейронов  $(q+1)$ -го слоя; однако возможен вариант соединения  $q$ -го слоя с произвольным  $(q+r)$ -м слоем.

Следует отметить, что классическим вариантом слоистых сетей являются сети прямого распространения (рис. 2).

3. *Сети с обратными связями.* Это сети, у которых информация с последующих слоев передается на предыдущие.

В качестве примера сетей с обратными связями на рис. 3 представлены так называемые *частично-рекуррентные сети* Элмана и Жордана.

Известные сети можно разделить по принципу структуры нейронов на *гомогенные* (или *однородные*) и *гетерогенные*. Гомогенные сети состоят из нейронов одного типа с единой функцией активации. В гетерогенную сеть входят нейроны с различными функциями активации.

Важно отметить существование *бинарных* и *аналоговых* сетей. Первые из них оперируют с двоичными сигналами, и выход каждого нейрона может принимать только два значения: логический ноль («заторможенное» состояние) и логическая единица («возбужденное» состояние).

Еще одна классификация делит нейронных сетей на *асинхронные* и *синхронные*. В первом случае в каждый момент времени свое состояние меняет лишь один нейрон. Во втором – состояние меняется сразу у целой группы нейронов, как правило, у всего слоя. Алгоритмически ход времени в нейронной сети

задается итерационным выполнением над нейронами одностипных действий. Далее будут рассматриваться только синхронные НС.

Сети можно классифицировать также по числу слоев.

Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуются НС. Чем сложнее НС, тем масштабнее задачи, подвластные ей.

Выбор структуры НС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные (на сегодняшний день) конфигурации. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами:

- возможности сети возрастают с увеличением числа ячеек сети, плотности связей между ними и числом выделенных слоев;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о так называемой динамической устойчивости сети;
- сложность алгоритмов функционирования сети (в том числе, на пример, введение нескольких типов синапсов – возбуждающих, тормозящих и др.) также способствует усилению мощи НС.

Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора, хотя в литературе приведены доказательства того, что для любого алгоритма существует нейронная сеть, которая может его реализовать.

**Лекция №2 Тема: Биологические основы функционирования нейрона**

Тематика искусственных нейронных сетей относится к междисциплинарной сфере знаний, связанных с биокибернетикой, электроникой, прикладной математикой, статистикой, автоматикой и даже с медициной.

Искусственные нейронные сети возникли на основе знаний о функционировании нервной системы живых существ. Они представляют собой попытку использования процессов, происходящих в нервных системах, для выработки новых технологических решений.

Нервная клетка, сокращенно называемая *нейроном*, является основным элементом нервной системы. Изучение механизмов функционирования отдельных нейронов и их взаимодействия принципиально важно для познания протекающих в нервной системе процессов поиска, передачи и обработки информации. С этой точки зрения представляется необходимым построить и изучить модель биологического нейрона.

Как и у любой другой клетки, у нейрона имеется тело со стандартным набором органелл, называемое *сомой*, внутри которого располагается ядро. Из сомы нейрона выходят многочисленные отростки, играющие ключевую роль в его взаимодействии с другими нервными клетками. Можно выделить два типа отростков: многочисленные тонкие, густо ветвящиеся *дендриты* и более толстый, расщепляющийся на конце *аксон* (рис. 1).

Входные сигналы поступают в клетку через *синапсы*, тогда как выходной сигнал отводится аксоном через его многочисленные нервные окончания, называемые *коллатералами*. Коллатералы контактируют с сомой и дендритами других нейронов, образуя очередные синапсы. Очевидно, что синапсы, подключающие к клетке выходы других нейронов, могут находиться как на дендритах, так и непосредственно на теле клетки.

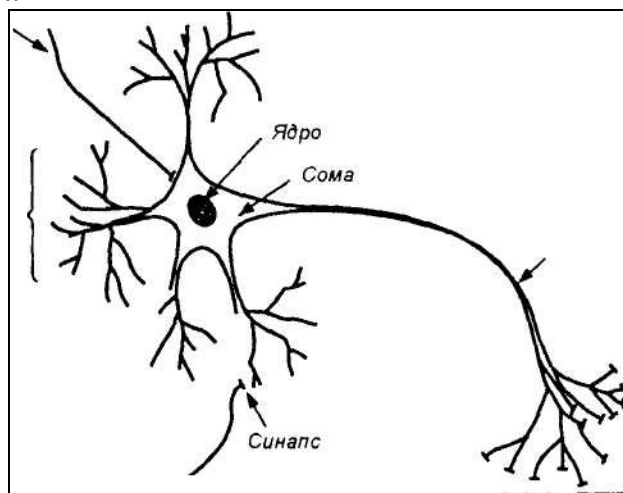
Передача сигналов внутри нервной системы - это очень сложный электрохимический процесс. С большим упрощением можно считать, что передача нервного импульса между двумя клетками основана на выделении особых химических субстанций, называемых *нейромедиаторами*, которые формируются под влиянием поступающих от синапсов раздражителей.



Эти субстанции воздействуют на клеточную мембрану, вызывая изменение ее энергетического потенциала, причем величина этого изменения пропорциональна количеству нейромедиатора, попадающего на мембрану.

*Аксон другого*

*нейрона*



*Дендриты*

*Аксон*

*Аксон другого нейрона*

*Концевые коллатералы*

Рис. 1. Упрощенная структура биологической нервной клетки

Синапсы отличаются друг от друга размерами и возможностями концентрации нейромедиатора вблизи своей оболочки. По этой причине импульсы одинаковой величины, поступающие на входы нервной клетки через различные синапсы, могут возбуждать ее в разной степени. Мерой возбуждения клетки считается уровень поляризации ее мембраны, зависящий от суммарного количества нейромедиатора, выделенного на всех синапсах.

Каждому входу клетки можно сопоставить численные коэффициенты (веса), пропорциональные количеству нейромедиатора, однократно выделяемого на соответствующем синапсе. В математической модели нейрона вход-

ные сигналы должны умножаться на эти коэффициенты для того, чтобы корректно учитывать влияние каждого сигнала на состояние нервной клетки. Синаптические веса должны быть натуральными числами, принимающими как положительные, так и отрицательные значения. В первом случае синапс оказывает возбуждающее, а во втором - тормозящее действие, препятствующее возбуждению клетки другими сигналами. Таким образом, действие возбуждающего синапса может моделироваться положительным значением синаптического веса, а действие тормозящего синапса - отрицательным значением.

В результате поступления входных импульсов на конкретные синапсы и высвобождения соответствующих количеств нейромедиатора происходит определенное электрическое возбуждение нервной клетки. Если отклонение от состояния электрического равновесия невелико либо если баланс возбуждений и торможений является отрицательным, клетка самостоятельно возвращается в исходное состояние, и на ее выходе какие-либо изменения не регистрируются. В этом случае считается, что уровень возбуждения клетки был ниже порога ее срабатывания. Если же сумма возбуждений и торможений превысила порог активации клетки, значение выходного сигнала начинает лавинообразно нарастать, принимая характерный вид нервного импульса, пересылаемого аксоном на другие нейроны, подключенные к данной клетке. Величина этого сигнала не зависит от степени превышения порога. Клетка действует по принципу "все или ничего".

После выполнения своей функции нейромедиатор удаляется. Механизм удаления заключается либо во всасывании этой субстанции клеткой, либо в ее разложении, либо в удалении за пределы синапса.

Одновременно с генерацией нервного импульса в клетке запускается процесс рефракции. Он проявляется как стремительное возрастание порога активации клетки до значения "плюс бесконечность", в результате чего сразу после генерации импульса нейрон теряет способность вырабатывать очередной сигнал даже при сильном возбуждении. Такое состояние сохраня-

ется в течение времени  $\Delta t_r$ , называемого периодом абсолютной рефракции. По окончании этого срока наступает период относительной рефракции  $\Delta t_w$ , за который порог срабатывания возвращается к первоначальному значению. В это время клетку можно активировать, но только с приложением более сильных возбуждений. В естественных процессах, как правило, выполняется отношение  $\Delta t_r \gg \Delta t_w$ .

Количество взаимодействующих друг с другом нервных клеток чрезвычайно велико. Считается, что человеческий мозг содержит около  $10^{11}$  нейронов, каждый из которых выполняет относительно примитивные функции суммирования весовых коэффициентов входных сигналов и сравнения полученной суммы с пороговым значением. Каждый нейрон имеет свои веса и свое пороговое значение. Они определяются местонахождением нейрона и решаемой им задачей и могут интерпретироваться аналогично содержимому локальной памяти процессора.

Громадное количество нейронов и межнейронных связей (до 1000 входов в каждый нейрон) приводит к тому, что ошибка в срабатывании отдельного нейрона остается незаметной в общей массе взаимодействующих клеток. Нейронная сеть проявляет высокую устойчивость к помехам - это "стабильная" сеть, в которой отдельные сбои не оказывают существенного влияния на результаты ее функционирования. Таково главное отличие нейронных систем от обычных электронных систем, созданных человеком. Следует подчеркнуть, что ни одна современная технология не позволяет построить искусственную нейронную сеть, близкую по масштабам к нейронной сети мозга. Однако изучение и копирование биологических нервных систем позволяют надеяться на создание нового поколения электронных устройств, имеющих аналогичные характеристики.

Другая важная особенность нервных систем - высокая скорость их функционирования, несмотря на относительно длительный цикл срабатывания каждой отдельной клетки, измеряемый в миллисекундах и показанный на рис. 2. Она достигается благодаря параллельной обработке информации в

мозге огромным количеством нейронов, соединенных многочисленными межнейронными связями. Такие операции, как распознавание образов и звуков либо принятие решений, выполняются человеческим мозгом за промежутки времени, измеряемые миллисекундами. Достижение такого результата при использовании полупроводниковой технологии VLSI все еще выходит за границы современных технических возможностей, хотя цикл срабатывания отдельных исполнительных элементов СБИС является достаточно коротким и имеет порядок  $10^{-8}$  с. Если удастся, взяв за образец нервную систему, создать устройство с высокой степенью параллельности выполнения независимых операций, то скорость его функционирования может быть существенно увеличена и приближена к уровню, наблюдаемому в процессах обработки информации биологическими объектам и могут интерпретироваться аналогично содержимому локальной памяти процессора.

*Лекция № 3 Тема: Первые модели нейронной сети. Функции активации.*

Исходя из биологических основ функционирования нейрона каждый нейрон можно считать своеобразным процессором. С каждым процессором (т.е. обрабатывающим элементом сети) связывается набор входящих связей, по которым к данному элементу поступают сигналы от других элементов сети, и набор исходящих связей, по которым сигналы данного элемента передаются другим элементам. Некоторые элементы предназначены для получения сигналов из внешней среды (и поэтому называются входными элементами), а некоторые — для вывода во внешнюю среду результатов вычислений (и поэтому такие элементы сети называются выходными элементами). В случае программного моделирования реальных процессов на входные элементы обычно подаются уже предварительно подготовленные данные из некоторого файла данных, а не от непосредственно связанных с внешней средой датчиков.

Процессор суммирует с соответствующими весами сигналы, приходящие от других нейронов, выполняет нелинейную (например, пороговую) решающую функцию и передает результирующее значение связанным с ним нейронам. В соответствии с действующим правилом "все или ничего" в

простейших моделях нейронов выходной сигнал принимает двоичные значения: 0 или 1. Значение 1 соответствует превышению порога возбуждения нейрона, а значение 0 — возбуждению ниже порогового уровня.

В одной из первых моделей нейрона, называемой моделью МакКаллока-Питса (предложенной в 1943 г.), нейрон считается бинарным элементом. Структурная схема этой модели представлена на рис. 1.

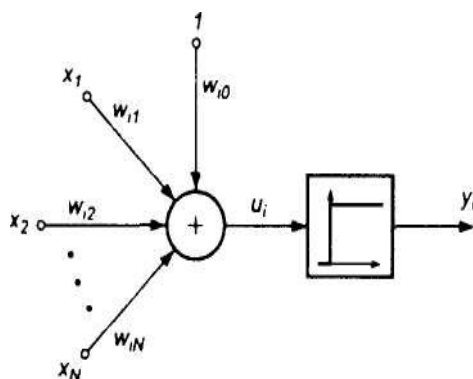


Рис. 1. Модель нервной клетки по МакКаллоку-Питсу

Входные сигналы  $x_j$  ( $j = 1, 2, \dots, N$ ) суммируются с учетом соответствующих весов  $w_{ij}$  (сигнал поступает в направлении от узла  $i$  к узлу  $j$ ) в сумматоре, после чего результат сравнивается с пороговым значением  $w_{i0}$ . Выходной сигнал нейрона  $y_i$ , определяется при этом зависимостью

$$y_i = f\left(\sum_{j=1}^N w_{ij}x_j(t) + w_{i0}\right). \quad (1)$$

Аргументом функции выступает суммарный сигнал  $u_i = \sum_{j=1}^N w_{ij}x_j(t) + w_{i0}$ .

Функция  $f(u_i)$  называется функцией активации. В модели МакКаллока-Питса это пороговая функция вида

$$f(u) = \begin{cases} 1 & \text{для } u > 0 \\ 0 & \text{для } u \leq 0 \end{cases} \quad (2)$$

Коэффициенты  $w_{ij}$ , присутствующие в формуле (1), представляют веса синаптических связей. Положительное значение  $w_{ij}$  соответствует возбуждающим синапсам, отрицательное значение  $w_{ij}$  — тормозящим синапсам, тогда как  $w_{ij} = 0$  свидетельствует об отсутствии связи между  $i$ -м и  $j$ -м нейронами.

Модель МакКаллока-Питса — это дискретная модель, в которой состояние нейрона в момент  $(t + 1)$  рассчитывается по значениям его входных сигналов в предыдущий момент  $t$ . Построение дискретной модели обосновывается проявлением рефракции у биологических нейронов, приводящей к тому, что нейрон может изменять свое состояние с конечной частотой, причем длительность периодов бездействия зависит от частоты его срабатывания.

Через несколько лет Д. Хебб в процессе исследования ассоциативной памяти предложил теорию обучения (подбора весов  $w_{ij}$ ) нейронов. При этом он использовал наблюдение, что веса межнейронных соединений при активации нейронов могут возрастать. В модели Хебба приращение веса  $\Delta w_{ij}$  в процессе обучения пропорционально произведению выходных сигналов  $y_i$  и  $y_j$  нейронов, связанных весом  $w_{ij}$ :

$$w_{ij}(k + 1) = w_{ij}(k) + \eta y_i(k)y_j(k), \quad (3)$$

где  $k$  означает номер цикла, а  $\eta$  - это коэффициент обучения.

В начале 60-х годов Б. Видроу предложил теоретическое обоснование и сформулировал принципы практической реализации адаптивных устройств обработки сигналов, что стало существенным вкладом в развитие нейронных сетей, функционирующих в режимах "онлайн" и "оффлайн".

В 1962 г. была опубликована книга Ф. Розенблатта, в которой представлена теория динамических нейронных систем для моделирования мозговой деятельности, основанная на перцептронной модели нервной клетки. В этой теории использовалось представление нейрона моделью МакКаллока-Питса, в которой функция активации принимала двоичные значения 0 и 1.

Ограниченные возможности одиночного перцептрона и составляемых из таких элементов одноуровневых сетей подверглись критике в книге М. Минского и С. Пейперта, что вызвало резкое снижение финансирования этой сферы научных исследований и привело в результате к замедлению развития искусственных нейронных сетей. Только отдельные научные группы, сконцентрированные во-

круг таких ученых, как Гроссберг, Видроу, фон дер Мальсбург, Амари, Фукушима и Кохонен, продолжали работу в этой области. И только бурное развитие в 80-х годах технологии производства полупроводниковых устройств сверхвысокой степени интеграции (VLSI) привело к резкому возрастанию интереса к средствам параллельной обработки информации, которыми считаются и искусственные нейронные сети. Начиная с опубликованных в 1982 г. работ Дж. Хопфилда, теория нейронных сетей развивается в стремительном темпе, а количество научных центров, занимающихся этой междисциплинарной сферой знаний, непрерывно увеличивается. Доработка или, точнее, повторное открытие принципа обратного распространения в применении к обучению многослойных сетей сняли те ограничения, которые стали главным объектом критики в книге М. Минского и С. Пейперта. Масштабное увеличение финансирования этой научной отрасли предопределило существенный прогресс как в теории, так и в практических приложениях. С учетом взрывного развития вычислительных систем это создало базу для реализации новых технологических решений в сфере технического распознавания образов, восприятия и объяснения, в управлении сложными системами, для обработки речевых сообщений и т.п. В настоящее время искусственные нейронные сети представляют собой высокоразвитую (особенно в теоретическом аспекте) отрасль знаний.

### **Правило вычисления сигнала активности**

Для всех элементов имеется правило вычисления выходного значения, которое предполагается передать другим элементам или во внешнюю среду (если речь идет о выходном элементе, представляющем конечный результат вычислений). Это правило называют функцией активности, а соответствующее выходное значение называют активностью соответствующего элемента. Активность может представляться либо некоторым действительным значением произвольного вида, либо действительным значением из некоторого ограниченного интервала значений (например, из интервала  $[0,1]$ ), или же некоторым значением из определенного дискретного набора значений (например,  $\{0, 1\}$  или

$\{+1,-1\}$ ). На вход функции активности поступает значение комбинированного ввода данного элемента. Примеры функций активности:

### 1) Тожественная функция

Функция активности для входных элементов может быть тождественной функцией, и это просто означает, что значение активности (сигнал, посылаемый другим элементам) оказывается в точности равным комбинированному вводу (рис. 4). Входные элементы обычно предназначены для распределения вводимых сигналов между другими элементами сети, поэтому для входных элементов обычно требуется, чтобы исходящий от элемента сигнал был таким же, как и входящий. В отличие от других элементов сети, входные элементы имеют только по одному входному значению. Например, каждый входной элемент может получать сигнал от одного соответствующего ему датчика, размещенного на фюзеляже самолета. Один этот элемент связывается со многими другими элементами сети, так что данные, полученные от одного датчика, оказываются распределенными между многими элементами сети. Поскольку входные элементы предназначены исключительно для того, чтобы распределять сигналы, получаемые из внешней среды, многие исследователи вообще не считают входные элементы частью нейронной сети.

### 2) Пороговая функция

В большинстве моделей нейронных сетей используются нелинейные функции активности. Пороговая функция ограничивает активность значениями 1 или 0 в зависимости от значения комбинированного ввода в сравнении с некоторой пороговой величиной 0 (рис. 5).

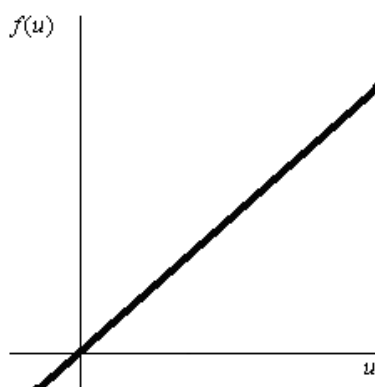
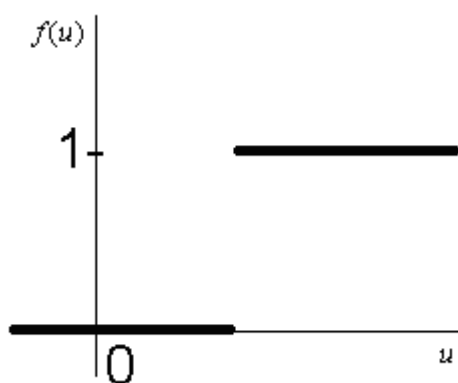




Рис. 4. Здесь активность в точности равна комбинированному вводу. Обратите внимание на то, что активность обозначается символом  $f(u)$

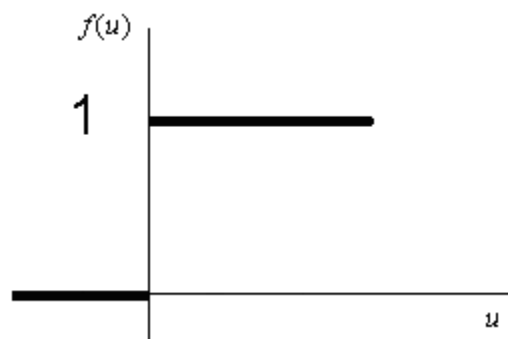


$$f(u) = \begin{cases} 1, & \text{если } u_j \geq 0, \\ 0, & \text{если } u_j < 0, \end{cases}$$

Рис. 5. Пороговая функция

Чаще всего удобнее вычесть пороговое значение (называемое смещением или сдвигом) из значения комбинированного ввода и рассмотреть пороговую функцию в ее математически эквивалентной форме, показанной на рис. 6. Сдвиг  $w_0$  данном случае оказывается отрицательным, а значение комбинирован-

ного ввода вычисляется по формуле  $u_j = w_0 + \sum_{i=1}^n x_i w_{ij}$ .



$$f(u) = \begin{cases} 1, & \text{если } u_j \geq 0, \\ 0, & \text{если } u_j < 0, \end{cases}$$

Рис. 6. Пороговая функция с учетным смещением

Сдвиг обычно интерпретируется как связь, исходящая от элемента, активность которого всегда равна 1. Комбинированный ввод в данном случае можно

представить в виде  $u_j = \sum_{i=0}^n x_i w_{ij}$ ,

где  $x_0$  всегда считается равным 1.

### 3) Сигмоидальная функция

Наиболее часто используемой функцией активности является сигмоидальная функция. Выходные значения такой функции непрерывно заполняют диапазон от 0 до 1. Примером может служить логистическая функция, показанная на рис. 7:  $f(u) = \frac{1}{1 + \exp(-u)}$

Наклон и область выходных значений логистической функции могут быть разными. Например, для биполярного сигмоида областью выходных значений является диапазон между -1 и 1.

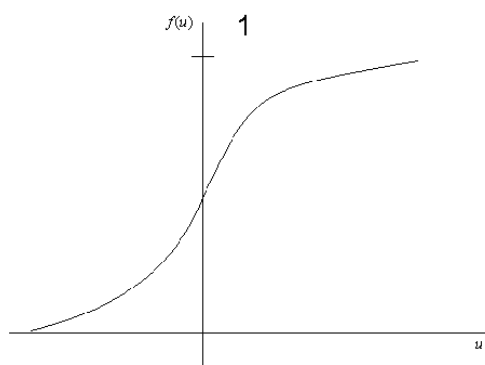


Рис. 7. Сигмоидальная функция

Лекция № 4 5 6 **Тема: Модели нейронов и методы их обучения.**

В соответствии с принципами функционирования биологических нейронов созданы различные математические модели, которыми в большей или меньшей степени реализуются свойства природной нервной клетки. Обобщенная схема, составляющая основу большинства таких моделей, восходит к представленной модели МакКаллока-Питса. Свойства нелинейной функции, особенно ее непрерывность, оказывают определяющее влияние на выбор способа обучения нейрона (подбор весовых коэффициентов). Другим важным фактором становится выбор стратегии обучения. Можно выделить два подхода: *обучение с учителем* и *обучение без учителя*.

При *обучении с учителем* предполагается, что, помимо входных сигналов, составляющих вектор  $x$ , известны также и ожидаемые выходные сигналы нейрона  $d_i$ , составляющие вектор  $d$ . В подобной ситуации подбор весовых коэффициентов должен быть организован так, чтобы фактические выходные сигнала

лы нейрона  $y_i$  принимали бы значения, как можно более близкие к ожидаемым значениям  $d_i$ . Ключевым элементом процесса обучения с учителем является знание ожидаемых значений  $d_i$ , выходного сигнала нейрона.

Если такой подход невозможен, остается выбрать стратегию обучения без учителя. Подбор весовых коэффициентов в этом случае проводится на основании либо конкуренции нейронов между собой (стратегии "*Winner Takes All - WTA*" (Победитель получает все) или "*Winner Takes Most - WTM*" (Победитель получает больше), либо с учетом корреляции обучающих и выходных сигналов (обучение по Хеббу). При *обучении без учителя* на этапе адаптации нейрона мы не можем прогнозировать его выходные сигналы, тогда как при обучении с учителем результат обучения predetermined заранее благодаря априори заданным обучающим выборкам.

### 1) Перцептрон

Простой перцептрон - это обычная модель МакКаллока-Питса с соответствующей стратегией обучения.

Весовые коэффициенты входов сумматора, на которые поступают входные сигналы  $x_j$ , обозначаются  $w_{ij}$ , а пороговое значение, поступающее с так называемого поляризатора, -  $w_{i0}$ . Нелинейная функция активации перцептрона представляет собой дискретную функцию ступенчатого типа, вследствие чего выходной сигнал нейрона может принимать только два значения - 0 или 1 в соответствии с правилом

$$y_i(u_i) = \begin{cases} 1 & \text{для } u \geq 0 \\ 0 & \text{для } u < 0 \end{cases}$$

(1)

где  $u_i$  обозначен выходной сигнал сумматора

$$u_i = \sum_{j=0}^N w_{ij} x_j \quad (2)$$

В приведенной формуле подразумевается, что имеющий длину  $N$  вектор  $x$  дополнен нулевым членом  $x_0 = 1$ , формирующим сигнал по-

ляризации,  $x = [x_0, x_1, \dots, x_N]$ . Обучение персептрона требует наличия учителя и состоит в таком подборе весов  $w_{ij}$ , чтобы выходной сигнал  $y_i$  был наиболее близок к заданному значению  $d_i$ . Это обучение гетероассоциативного типа, при котором каждой обучающей выборке, представляемой вектором  $x$  априори поставлено в соответствие ожидаемое значение  $d_i$  на выходе  $i$ -го нейрона.

## 2) Сигмоидальный нейрон

Нейрон сигмоидального типа (рис. 2) имеет структуру, подобную модели МакКаллока-Питса, с той разницей, что функция активации является непрерывной и может быть выражена в виде сигмоидальной униполярной или биполярной функции. Униполярная функция, как правило, представляется формулой

$$f(x) = \frac{1}{1 + e^{-\beta x}}, \quad (7)$$

тогда как биполярная функция задается в виде

$$f(x) = th(\beta x) = \tanh(\beta x) = \frac{e^{\beta x} - e^{-\beta x}}{e^{\beta x} + e^{-\beta x}} \quad (8)$$

В этих формулах параметр  $\beta$  подбирается пользователем. Его значение влияет на форму функции активации. На рис. 3 представлены графики сигмоидальной функции от переменной  $x$  для различных значений  $\beta$

При малых величинах  $\beta$  график функции достаточно пологий, но по мере роста значения  $\beta$  крутизна графика увеличивается. При  $\beta \rightarrow \infty$  сигмоидальная функция превращается в функцию ступенчатого типа, идентичную функции активации персептрона. На практике чаще всего для упрощения используется значение  $\beta = 1$ .

## 3) Нейрон типа "адалайн"

Модель нейрона типа "адалайн" (англ.: *ADaptive Linear NEuron* - адаптивный линейный нейрон) была предложена Б. Видроу. По методу весового суммирования сигналов нейрон типа "адалайн" аналогичен представленным ранее моделям нейронов. Функция активации имеет тип *signum*, т.е.

$$y_i(u_i) = \begin{cases} 1 & \text{для } u_i > 0 \\ -1 & \text{для } u_i \leq 0 \end{cases} \quad (18)$$

Адаптивный подбор весовых коэффициентов осуществляется в процессе минимизации квадратичной ошибки, определяемой как

$$E(w) = \frac{1}{2} e_i^2 = \frac{1}{2} \left[ d_i - \sum_{j=0}^N w_{ij} x_j \right]^2 \quad (19)$$

Следует обратить внимание, что, несмотря на нелинейный характер модели, в целевой функции присутствуют только линейные члены, представляющие собой сумму взвешенных входных сигналов. В связи с выполнением условия непрерывности целевой функции стало возможным применение алгоритма градиентного обучения. Как и в ситуации с сигмоидальным нейроном, в алгоритме Видроу для минимизации целевой функции применяется метод наискорейшего спуска. Значения весовых коэффициентов могут уточняться либо дискретным способом

$$w_{ij}(t+1) = w_{ij}(t) + \eta e_i x_j, \quad (20)$$

либо аналоговым способом - путем решения разностных уравнений вида

$$\frac{dw_{ij}}{dt} = \mu e_i x_j \quad (21)$$

в которых в соответствии с зависимостью (19)  $e_i = \left( d_i - \sum_{j=0}^N w_{ij} x_j \right)$ . Несмотря на

то, что адалайн имеет на выходе нелинейный блок типа `signum`, он все же считается линейным элементом, поскольку в определении целевой функции нелинейности отсутствуют, а подбор весов происходит так, как будто никакой нелинейности не существует.

Нейрон типа "адалайн" имеет относительно простую практическую реализацию как в случае аналогового подхода на основе уравнения (21), так и в дискретном варианте на базе выражения (20). Основные компоненты модели в первом случае - это вычислительные элементы (интеграторы и сумматоры), тогда как во втором случае - это элементы задержки, описываемые оператором запаздывания

$z^{-1}$ , и также интеграторы и сумматоры. Обе адалайн-модели могут служить базой для компьютерного моделирования нейрона этого типа.

В практических приложениях нейроны типа "адалайн" всегда используются группами, образуя слои, называемые *мадалайн* (англ.: *Many adaline* - много адалайн). Каждый входящий в слой нейрон обучается по правилу адалайн. Выходные сигналы отдельных нейронов такого слоя могут формироваться различными способами. Б. Видроу предложил три базовых типа межнейронных соединений: OR, AND и мажоритарное. На рис. 6 а, б и в показаны схемы таких соединений.

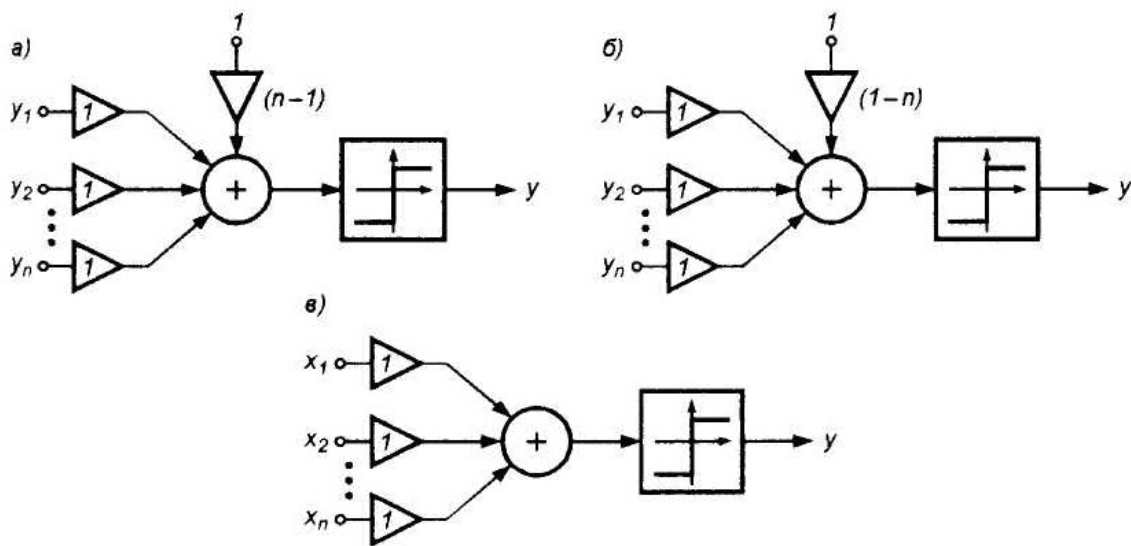


Рис. 6. Сеть мадалайн с выходами типа: а) OR; б) AND; в) мажоритарный

Конкретные сигналы  $y_i$  суммируются с учетом порогового значения, устанавливаемого отдельно для каждого типа связи. Для схемы OR порог имеет значение  $(n-1)$ , для схемы AND - значение  $(1-n)$ , а для мажоритарной схемы - нулевое значение. Благодаря применению функции активации типа *signum* выходной сигнал  $y$  принимает значение  $+1$ , когда хотя бы один из входных сигналов имеет значение  $+1$  (OR), когда все входные сигналы  $y_i$  имеют значения  $+1$  (AND) либо когда большинство сигналов  $y_i$  имеет значение  $+1$  (мажоритарное соединение).

#### 4) Инстар и оутстар Гроссберга

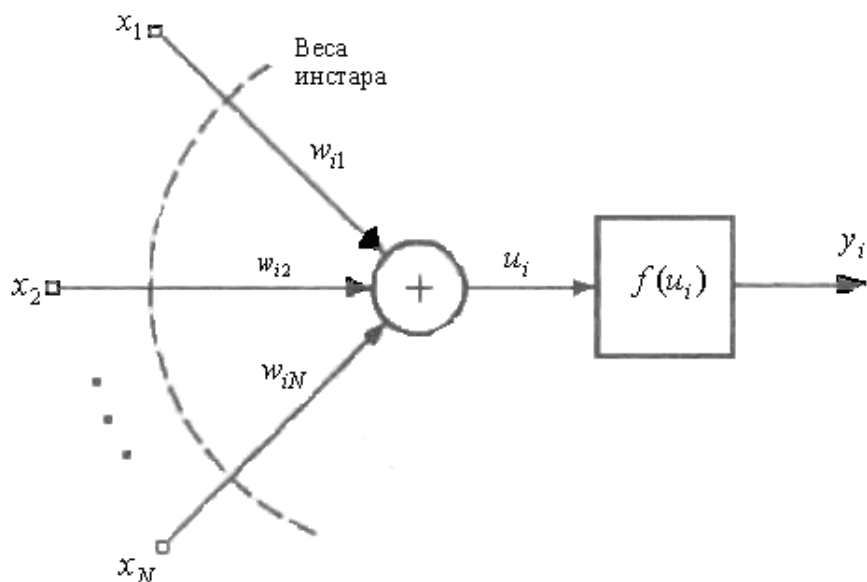


Рис. 7. Структурная схема инстара

Нейроны типа инстар и оутстар - это взаимодополняющие элементы. Инстар адаптирует веса сигналов, поступающих на сумматор нейрона, к своим входным сигналам, а оутстар согласовывает веса выходящих из нейрона связей с узлами, в которых формируются значения выходных сигналов. Нейрон типа инстар был определен С. Гроссбергом. На рис. 7. представлена структурная схема инстара.

Сигналы  $x_j$ , подаваемые с весовыми коэффициентами  $w_{ij}$   $i$ -го инстара, суммируются в соответствии с выражением

$$u_i = \sum_{j=1}^N w_{ij} x_j . \quad (22)$$

В соответствии с функцией активации на выходе нейрона вырабатывается выходной сигнал  $y_i = f(u_i)$ . Часто в инстаре применяется линейная форма функции активации, и тогда  $y_i = u_i$ . Обучение инстара (подбор весов  $w_{ij}$ ) производится по правилу Гроссберга, в соответствии с которым

$$w_{ij}(t+1) = w_{ij}(t) + \eta y_i [x_j - w_{ij}(t)] , \quad (23)$$

где  $\eta$  - это коэффициент обучения, значение которого, как правило, выбирается из интервала (0,1). Входные данные, представляемые в виде вектора  $x$ , выражены чаще всего в нормализованной форме, в которой  $\|x\| = 1$ . Нормализация компонентов вектора  $x$  выполняется по формуле

$$x_j \leftarrow \frac{x_j}{\sqrt{x_1^2 + x_2^2 + \dots + x_N^2}}. \quad (24)$$

Результаты обучения по методу Гроссберга в значительной степени зависят от коэффициента обучения  $\eta$ . При выборе  $\eta = 1$  веса  $w_{ij}$  становятся равными значениям  $x_j$  уже после первой итерации. Ввод очередного входного вектора  $x$  вызовет адаптацию весов к новому вектору и абсолютное "забывание" предыдущих значений. Выбор  $\eta < 1$  приводит к тому, что в результате обучения весовые коэффициенты  $w_{ij}$  принимают усредненные значения обучающих векторов  $x$ .

Допустим, что  $i$ -й инстар был обучен на некотором нормализованном относительно своих компонентов входном векторе  $x_1$ . В этом случае на векторе весов инстара выполняется отношение:  $w = [w_{i1}, w_{i2}, \dots, w_{iN}]^T = x_1$ . В режиме классификации при вводе очередного входного вектора  $x_2$  инстар вырабатывает сигнал  $u_i$  вида

$$u_i = w^T x_2 = x_1^T x_2 = \|x_1\| \|x_2\| \cos \varphi_{12}. \quad (25)$$

Вследствие нормализации амплитуд входных векторов получаем:

$$u_i = \cos \varphi_{12}. \quad (26)$$

При выполнении условия  $x_2 = x_1$  реакция инстара будет равна  $u_i = 1$ . В случае, когда входные векторы отличаются друг от друга, реакция инстара будет пропорциональна косинусу угла между этими векторами. Для ортогональных векторов  $u_i = 0$ .

В итоге натренированный инстар функционирует как векторный классификатор, сопоставляющий очередной поданный на его вход вектор с вектором, сформированным в процессе обучения. В случае максимального совпадения этих векторов реакция инстара будет максимальной (наиболее близкой к единице). Если инстар обучался на группе достаточно похожих векторов с коэффициентом обучения  $\eta < 1$ , то его весовые коэффициенты примут значения, усредненные по этим векторам, и в ре-



жиме классификации он будет лучше всего реагировать на входные векторы, параметры которых наиболее близки к средним значениям векторов, входивших в обучающую группу.

Инстар может обучаться как с учителем, так и без него. Во втором случае в правиле Гроссберга в качестве значения  $y_i$  принимается фактическое значение выходного сигнала инстара. При обучении с учителем значение  $y_i$  заменяется ожидаемым значением  $d_i$ , т.е.  $y_i = d_i$ .

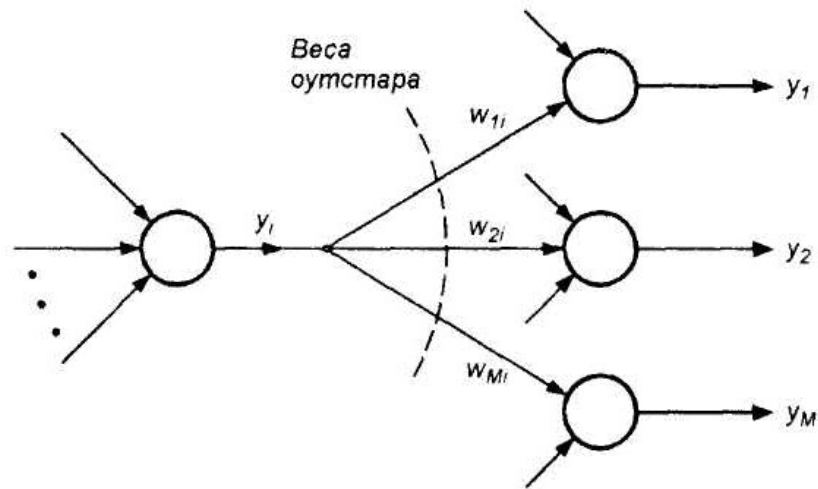


Рис. 8. Структурная схема оутстара

Нейрон типа оутстар Гроссберга представляет собой комплементарное дополнение инстара. Если инстар обучается с целью распознавать вектор, подаваемый на его вход, то оутстар должен генерировать вектор, необходимый связанным с ним нейронам. Структурная схема оутстара представлена на рис. 8.  $i$ -й нейрон-источник высылает свой выходной сигнал  $y_i$  взаимодействующим с ним нейронам, выходные сигналы которых обозначены  $y_j$  ( $j = 1, 2, \dots, M$ ). Оутстар, как правило, является линейным нейроном. Обучение состоит в таком подборе его весов  $w_{ij}$ , чтобы выходные сигналы оутстара были равны ожидаемым значениям  $y_j$  взаимодействующих с ним нейронов. Обучение оутстара согласно правилу Гроссберга проводится в соответствии с выражением

$$w_{ji}(t+1) = w_{ji}(t) + \eta y_i [y_j - w_{ji}(t)], \quad (27)$$

в котором  $\eta$  - это коэффициент обучения, а  $y_i$  - выходной сигнал  $i$ -го нейрона, выступающего в роли источника. Зависимость (27) для оутстара аналогична выражению (23), по которому обучается инстар. В режиме распознавания в момент активизации нейрона-источника оутстар будет генерировать сигналы, соответствующие ожидаемым значениям  $y_j$ .

Нейроны типа инстар и оутстар существенным образом отличаются от нейронов трех типов, определенных ранее. Основу обучения персептрона, сигмоидального нейрона и адалайна составляет пара обучающих векторов  $(x, d)$ . Они могут обучаться только с учителем. При обучении инстара и оутстара весовые коэффициенты подстраиваются под входные или выходные векторы. Обучение может проводиться как с учителем, так и без него.

### 5) Нейроны типа WTA

Нейроны типа WTA (англ.: *Winner Takes All* - Победитель получает все) имеют входной модуль в виде стандартного сумматора, рассчитывающего сумму входных сигналов с соответствующими весами  $w_{ij}$ . Выходной сигнал  $i$ -го сумматора определяется согласно формуле

$$u_i = \sum_{j=0}^N w_{ij} x_j \quad (28)$$

Группа конкурирующих между собой нейронов (рис. 9) получает одни и те же входные сигналы  $x_j$ . В зависимости от фактических значений весовых коэффициентов суммарные сигналы  $u_i$  отдельных нейронов могут различаться. По результатам сравнения этих сигналов победителем признается нейрон, значение  $u_i$  у которого оказалось наибольшим. Нейрон-победитель вырабатывает на своем выходе состояние 1, а остальные (проигравшие) нейроны переходят в состояние 0.

Для обучения нейронов типа WTA не требуется учитель, оно протекает аналогично обучению инстара, с использованием нормализованных входных векторов  $x$ . На начальном этапе случайным образом выбираются

весовые коэффициенты каждого нейрона, нормализуемые относительно 1. После подачи первого входного вектора  $x$  определяется победитель этапа. Победивший в этом соревновании нейрон переходит в состояние 1, что позволяет ему провести уточнение весов его входных линий  $w_{ij}$  (по правилу Гроссберга).

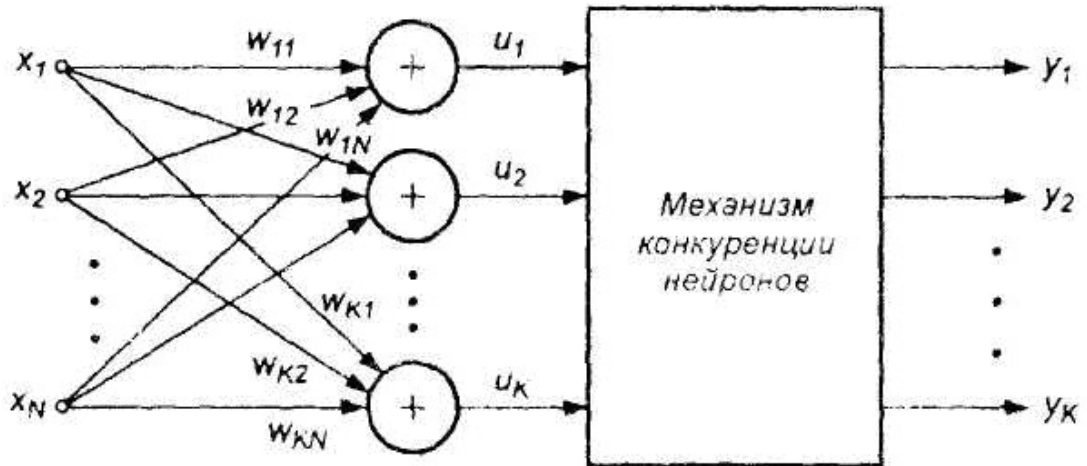


Рис. 9. Схема соединения нейронов типа WTA

Проигравшие нейроны формируют на своих выходах состояние 0, что блокирует процесс уточнения их весовых коэффициентов. Вследствие бинарности значения выходных сигналов конкурирующих нейронов (0 или 1) правило Гроссберга может быть несколько упрощено:

$$w_{ij}(t+1) = w_{ij}(t) + \eta [x_j - w_{ij}(t)] \quad (29)$$

На функционирование нейронов типа WTA оказывает существенное влияние нормализация входных векторов и весовых коэффициентов. Выходной сигнал  $u_i$   $i$ -го нейрона в соответствии с формулой (25) может быть описан векторным отношением

$$u_i = w^T x = \|w\| \|x\| \cos \varphi_i. \quad (30)$$

Поскольку  $\|w\| = \|x\| = 1$ , значение  $u_i$  определяется углом между векторами  $x$  и  $w$ ,  $u_i = \cos \varphi_i$ . Поэтому победителем оказывается нейрон, вектор весов которого оказывается наиболее близким текущему обучающему вектору  $x$ . В результате победы нейрона уточняются его весовые коэффициенты, значения которых приближаются к значениям текущего обучающего векто-

ра  $x$ . Если на вход сети будет подаваться множество близких по значениям векторов, побеждать будет один и тот же нейрон. Поэтому его веса станут равными усредненным значениям тех входных векторов, благодаря которым данный нейрон оказался победителем. Проигравшие нейроны не изменяют свои веса. Только победа при очередном представлении входного вектора позволит им произвести уточнение весовых коэффициентов и продолжить процесс обучения в случае еще одной победы.

Следствием такой конкуренции становится самоорганизация процесса обучения. Нейроны уточняют свои веса таким образом, что при предъявлении группы близких по значениям входных векторов победителем всегда оказывается один и тот же нейрон. В процессе функционирования именно этот нейрон благодаря соперничеству распознает свою категорию входных данных. Системы такого типа чаще всего применяются для классификации векторов.

В качестве примера рассмотрим нейронную сеть, состоящую из четырех нейронов типа WTA и предназначенную для классификации входных двухкомпонентных векторов (рис. 10). Входные обучающие векторы  $x$  представлены в нормализованной форме.

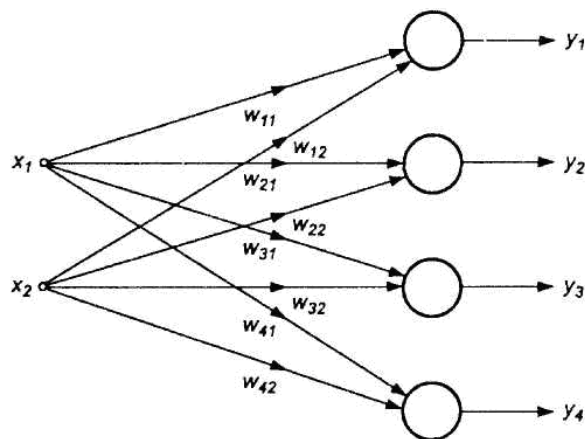


Рис. 10. Нейронная сеть типа WTA

Серьезной проблемой при обучении WTA остается проблема мертвых нейронов, которые после инициализации ни одного раза не победили в конкурентной борьбе и остались в состоянии, сформированном в начальный момент времени. Каждый мертвый нейрон уменьшает эффективное количество эле-

ментов, прошедших обучение, и соответственно увеличивает общую погрешность распознавания данных. Для разрешения этой проблемы применяется модифицированное обучение, основанное на учете прошлых побед каждого нейрона и штрафования (временной дисквалификации) тех из них, которые побеждали чаще всего. Дисквалификация слишком активных нейронов может осуществляться либо назначением порогового числа побед, по достижении которого наступает обязательная пауза, либо уменьшением фактического значения  $u_i w$ , при нарастании количества побед  $i$ -го нейрона.

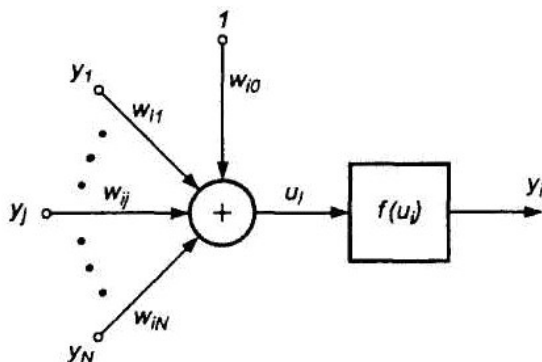
### б) Модель нейрона Хебба

Д. Хебб в процессе исследования нервных клеток заметил, что связь между двумя клетками усиливается, если обе клетки пробуждаются (становятся активными) в один и тот же момент времени. Если  $j$ -я клетка с выходным сигналом  $y_j$  связана с  $i$ -й клеткой, имеющей выходной сигнал  $y_i$ , связью с весом  $w_{ij}$ , то на силу связи этих клеток влияют значения выходных сигналов  $y_i$  и  $y_j$ .

Д. Хебб предложил формальное правило, в котором отразились результаты его наблюдений. В соответствии с правилом Хебба, вес  $w_{ij}$  нейрона изменяется пропорционально произведению его входного и выходного сигналов

$$\Delta w_{ij} = \eta y_j y_i, \quad (31)$$

где  $\eta$  - это коэффициент обучения, значение которого выбирается в интервале (0,1). Правило Хебба может применяться для нейронных сетей различных типов с разнообразными функциями активации моделей отдельных нейронов.



Структурная схема нейрона Хебба, представленная на рис. 11, соответствует стандартной форме модели нейрона. Связь с весом  $w_{ij}$ , способ подбора значения которого задается отношением (31), соединяет входной сигнал  $y_j$  с сумматором

Рис. 11. Структурная схема нейрона Хебба  $i$ -го нейрона, вырабатывающего выходной сигнал  $y_i$ . Обучение нейрона по правилу Хебба

может проводиться как с учителем, так и без него. Во втором случае в правиле Хебба используется фактическое значение  $y_i$  выходного сигнала нейрона. При обучении с учителем вместо значения выходного сигнала  $y_i$  используется ожидаемая от этого нейрона реакция  $d_i$ . В этом случае правило Хебба записывается в виде

$$\Delta w_{ij} = \eta y_j d_i. \quad (32)$$

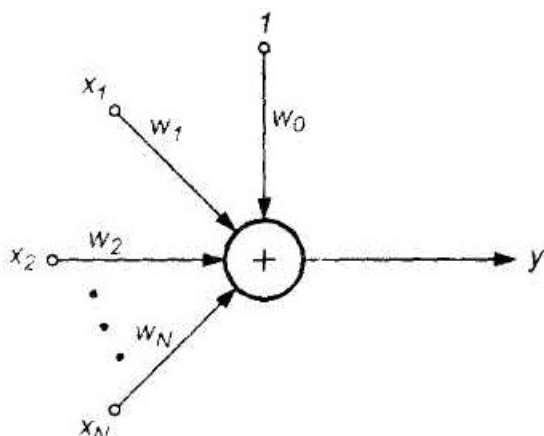
Правило Хебба характеризуется тем, что в результате его применения веса могут принимать произвольно большие значения, поскольку в каждом цикле обучения происходит суммирование текущего значения веса и его приращения  $\Delta w_{ij}$ :

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}. \quad (33)$$

Один из способов стабилизации процесса обучения по правилу Хебба состоит в учете для уточнения веса последнего значения  $w_{ij}$ , уменьшенного на коэффициент забывания  $\gamma$ . При этом правило Хебба представляется в виде

$$w_{ij}(t+1) = w_{ij}(t)(1 - \gamma) + \Delta w_{ij} \quad (34)$$

Значение коэффициента забывания  $\gamma$  выбирается, как правило, из интервала  $(0, 1)$  и чаще всего составляет некоторый процент от коэффициента обучения  $\eta$ . Применение больших значений  $\gamma$  приводит к тому, что нейрон забывает значительную часть того, чему он обучился в прошлом. Рекомендуемые значения коэффициента забывания -  $\gamma < 0,1$ , при которых нейрон сохраняет большую часть информации, накопленной в процессе обучения, и получает возможность стабилизировать значения весов на определенном уровне.



При обучении линейного нейрона по правилу Хебба стабилизация не происходит даже при вводе коэффициента забывания.

Выходной сигнал нейрона, структурная

схема которого приведена на рис. 13, определяется выражением

$$y = \sum_j w_j x_j = w^T x = x^T w \quad (35)$$

Рис. 12. Модель линейного нейрона

Хебба

Если согласно правилу Хебба

$$\Delta w = \eta x y \quad (36)$$

подставить выражение (35) в формулу (36) и выбрать для упрощения  $\eta = 1$ , то получим приращение вектора весов  $\Delta w$  в виде

$$\Delta w = C w, \quad (37)$$

где  $C = x x^T$  - это матрица корреляции, которая по определению является симметричной и положительно полуопределенной и, следовательно, имеет собственные натуральные и неотрицательные значения. При выполнении операций, описываемых зависимостью (37) и повторяемых на положительно полуопределенной матрице  $C$ , процесс становится расходящимся, а значения компонентов вектора  $w$  стремятся к бесконечности.

Нестабильность правила Хебба в процессе обучения можно устранить ограничением вектора весов за счет операции ренормализации, т.е. таким подбором пропорционального коэффициента  $\alpha$  на каждом шаге обучения, чтобы  $w' = \alpha w$  при  $\|w'\| = 1$ . Этот метод достаточно сложен и требует дополнительных трудозатрат на этапе обучения.

Е. Ойя модифицировал правило Хебба таким образом, что и без ренормализации процесса обучения вектор весов самостоятельно стремится к  $\|w\| = 1$ . В соответствии с правилом Ойи уточнение весов производится согласно выражению

$$\Delta w = \eta y (x_i - u w_i). \quad (38)$$

Это правило напоминает обратное распространение, поскольку сигнал  $x_i$  модифицируется обратным сигналом, связанным с выходным сигналом  $u$  нейрона. Для каждого отдельно взятого нейрона правило Ойя может считаться ло-

кальным, так как в процессе модификации  $x_i$  принимается во внимание только тот весовой коэффициент, значение которого подбирается в текущий момент времени.

Доказательство ограниченности весов, уточняемых по правилу Ойя, можно получить, заменяя скалярное выражение (38) векторной формой, которая с учетом упрощения  $\eta = 1$  и в соответствии с (38) приобретает вид:

$$\Delta w = C w - (w^T C w) w \quad (39)$$

Стабильность процесса обучения достигается, когда при достаточно длительном обучении обеспечивается  $\|\Delta w\| = 0$ , т.е.

$$C w = (w^T C w) w. \quad (40)$$

Если собственное значение корреляционной матрицы  $C$  обозначить  $\lambda$ , а вектор  $w$  подбирать как связанный с ней собственный вектор, то по определению собственного значения имеем  $C w = \lambda w$ . Подставляя это выражение в формулу (39), получаем:

$$\lambda = w^T C w = w^T \lambda w = \lambda |w|^2. \quad (41)$$

Из (41) следует, что применение для обучения модифицированного правила Хебба приводит к ограничению модуля вектора  $w$  единицей  $|w| = 1$ , обеспечивающему ограниченность значений весовых коэффициентов.

### 7) Стохастическая модель нейрона

В отличие от всех детерминированных моделей, определенных ранее, в стохастической модели выходное состояние нейрона зависит не только от взвешенной суммы входных сигналов, но и от некоторой случайной переменной, значения которой выбираются при каждой реализации из интервала  $(0,1)$ .

В стохастической модели нейрона выходной сигналу, принимает значения

$$\pm 1 \text{ с вероятностью } \text{Pr ob}(y_i = \pm 1) = \frac{1}{1 + \exp(\pm 2\beta u_i)} \text{ где } u_i \text{ обозначена взвешенная}$$

сумма входных сигналов  $i$ -го нейрона, а  $\beta$  — это положительная константа,



чаще всего равная 1. Процесс обучения нейрона в стохастической модели состоит из следующих этапов:

- Расчет взвешенной суммы  $u_i = \sum_{j=0}^N w_{ij} x_j$  для каждого нейрона сети.
- Расчет вероятности того, что  $y_i$  принимает значение  $\pm 1$  в соответствии с формулой

$$\text{Prob}(y_i = \pm 1) = \frac{1}{1 + \exp(\pm 2\beta u_i)}$$

(42)

- Генерация значения случайной переменной  $R \in (0,1)$  и формирование выходного сигнала  $y_i = \pm 1$ , если  $R < \text{Prob}(y_i = \pm 1)$  или  $y_i = \mp 1$ , в противном случае.
- Определенный таким образом процесс осуществляется на случайно выбранной группе нейронов, вследствие чего их состояние модифицируется в соответствии с предложенным правилом.
- После фиксации состояния отобранных нейронов их весовые коэффициенты модифицируются по применяемому правилу уточнения весов. Например, при обучении с учителем по правилу Видроу-Хоффа адаптация весов проводится по формуле

$$\Delta w_{ij} = \eta x_j (d_i - y_i). \quad (43)$$

Такой способ подбора весов приводит в результате к минимизации целевой функции, определенной как среднеквадратичная погрешность

$$E = \frac{1}{2} \sum_{k=1}^p \sum_{i=1}^n \left( d_i^{(k)} - y_i^{(k)} \right)^2,$$

рассчитываемая по всем  $n$  нейронам и  $p$  обучающим выборкам.

### Лекция № 7 *Тема: Ассоциация образцов*

Рассмотрим случай запоминаемых пар образцов. Идея заключается в том, чтобы выбрать нужный образец из памяти, даже если у нас нет всей необходимой информации для начала поиска сохраненного образца. Например, вы хотите найти книгу в библиотеке, но не помните ее названия. При этом если

вы знаете имя автора и описание того, чему книга посвящена, этого уже достаточно (с большой долей уверенности!), чтобы найти ассоциируемый с этой информацией объект.

Когда сохраняемая в памяти пара ассоциируемых образцов создается одинаковыми образцами, память называется автоассоциативной, а если образцы являются разными, то память называется гетероассоциативной. В этой главе будут рассмотрены три модели нейронных сетей для автоассоциации образцов.

### *Дискретная сеть Хопфилда*

Сеть Хопфилда (Hopfield) является автоассоциативной сетью, ведущей себя подобно памяти, которая может вспомнить сохраненный образец даже по подсказке (в виде вводимых данных), представляющей собой искаженную по мехам версию нужного образца. Например, сеть может сохранить набор изображений букв, а когда сети будет представлена искаженная версия сохраненного символа, сеть должна оказаться способной найти истинный экземпляр. Дискретная сеть Хопфилда имеет следующие характеристики.

- Один слой элементов (входные элементы, представляющие входной образец, не учитываются).
- Каждый элемент связывается со всеми другими элементами, но элемент не связывается с самим собой.
- За один шаг обновляется только один элемент, в отличие, например, от сети с обратным распространением ошибок, где все элементы слоя могут изменяться одновременно, если сеть реализована в виде аппаратных средств с соответствующими параллельными возможностями.
- Элементы обновляются в случайном порядке, но в среднем каждый элемент должен обновляться в одной и той же мере. Например, в случае сети из 10 элементов после 100 обновлений каждый элемент должен обновиться приблизительно 10 раз.
- Вывод элемента ограничен значениями 0 или 1.

Сеть Хопфилда является рекуррентной в том смысле, что для каждого входного образца выход сети повторно используется в качестве ввода до тех

пор, пока не будет достигнуто устойчивое состояние. Пример сети Хопфилда показан на рис. 1. Удобно считать, что сеть Хопфилда не имеет входных элементов, так как входной вектор просто определяет начальные значения активности элементов. Например, если ввод является двоичным, то входной вектор [1 1 0 1] означает, что значения активности для элементов {1, 2, 4} будут равны 1, а для элемента {3} активность будет равна 0. Элемент обновляется тогда, когда все элементы передадут свои значения активности по имеющимся взвешенным связям, после чего вычисляется сумма произведений (т.е. берется скалярное произведение). Значение активности элемента получается на основе использования некоторого правила активизации. Каждый элемент сети Хопфилда имеет состояние, характеризующееся значением активности, которое должен посылать данный элемент другим элементам, а состояние сети в любой момент времени задается вектором состояний всех ее элементов.

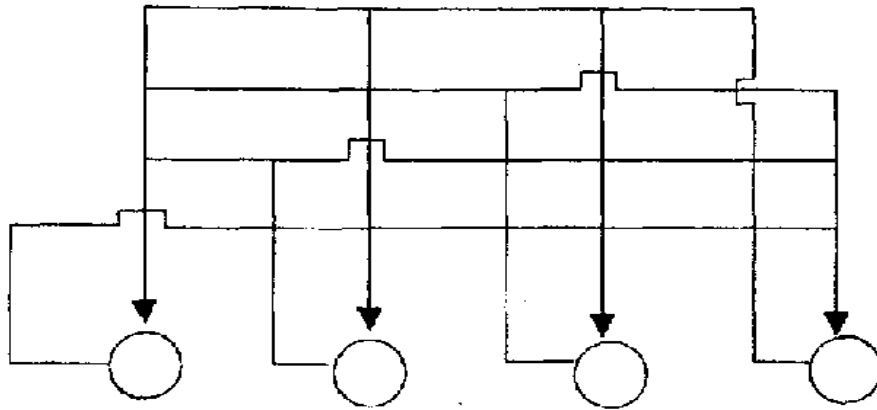


Рис. 1. Сеть Хопфилда с четырьмя элементами. Для каждого элемента входного вектора имеется свой элемент сети. Элементы сети связаны со всеми остальными ее элементами, но не сами с собой. Связи являются двунаправленными

В качестве входных данных сети Хопфилда можно использовать двоичные, но здесь мы будем использовать +1 для обозначения состояния "включено" и -1 — для состояния "выключено". Комбинированный ввод элемента вычисляется по формуле:

$$net_i = \sum_{j=1}^n s_j w_{ij} ,$$

где  $s_i$  обозначает состояние элемента с номером  $i$ . Когда элемент обновляется, его состояние изменяется в соответствии с правилом

$$s_j = \begin{cases} +1, & \text{если } net_j > 0 \\ -1, & \text{если } net_j < 0 \end{cases}$$

Эта зависимость называется сигнум-функцией и в более краткой форме она записывается в виде

$$s_j = \text{sgn}(net_j).$$

Если комбинированный ввод оказывается равным нулю, то элемент остается в состоянии, в котором он пребывал перед обновлением.

Сеть работает очень просто. Входной вектор задает начальные состояния всех элементов. Элемент для обновления выбирается случайным образом. Выбранный элемент получает взвешенные сигналы от всех остальных элементов и изменяет свое состояние. Выбирается другой элемент, и процесс повторяется. Сеть достигает предела, когда ни один из ее элементов, будучи выбранным для обновления, не меняет своего состояния.

Весовые значения для сети Хопфилда определяются непосредственно из учебных данных без необходимости проведения обучения в более привычном смысле. Сеть Хопфилда ведет себя как память, и процедура сохранения отдельного вектора представляет собой вычисление прямого произведения вектора с ним самим. В результате этой процедуры создается матрица, задающая весовые значения для сети Хопфилда, в которой все диагональные элементы должны быть установлены равными нулю (поскольку диагональные элементы задают автосвязи элементов, а элементы сами с собой не связаны). Таким образом, весовая матрица, соответствующая сохранению вектора  $x$ , задается формулой  $W = x^T x$ .

### ***Функция энергии***

В 1984 году Хопфилд доказал, что его сеть должна сходиться к устойчивому набору значений активности, рассмотрев функцию энергии системы. В представленной нами форме сеть определяет необходимость изменения состояния элемента по пороговому значению, равному нулю, поэтому мы можем ис-

пользовать упрощенную версию функции энергии Хопфилда, подобную той, которая приводится в [Haykin, 1994]:

$$E = -\frac{1}{2} \sum_j \sum_i s_j s_i w_{ij} .$$

Если элемент  $j$  изменяет свое состояние на  $\Delta s_j$ , то изменение энергии будет равно

$$\Delta E = -\Delta s_j \sum_i s_i w_{ij} .$$

Можно рассматривать изменение энергии как функцию  $\Delta s_j$  и  $\sum_i s_i w_{ij}$ .

Таблица 1. Изменения энергии при изменениях состояния элемента

Старое $s_j$	$\sum_i s_i w_{ij}$	Новое $s_j$	$\Delta s_j$	Изменение энергии
Положительное	Положительное	Положительное	Положительное	Отрицательное
Положительное	Отрицательное	Отрицательное	Отрицательное	Отрицательное
Отрицательное	Отрицательное	Отрицательное	Отрицательное	Отрицательное
Отрицательное	Положительное	Положительное	Положительное	Отрицательное

Из табл. 1 видно, что  $\Delta s_j$  и  $\sum_i s_i w_{ij}$  имеют одинаковые знаки, поэтому энергия при переходе от итерации к итерации всегда уменьшается. Например, первая строка данных таблицы говорит о том, что если элемент находится в положительном состоянии и его комбинированный ввод больше нуля (т.е. положителен), то новое состояние останется положительным и изменение состояния тоже будет положительным, а изменение энергии таким образом будет отрицательным.

### *Двунаправленная ассоциативная память*

Сетью, имеющей много общего с сетью Хопфилда, является двунаправленная ассоциативная память (сеть ВАМ — Bidirectional Associate Memory), предложенная Коско. Сеть ВАМ является гетероассоциативной рекуррентной сетью. Сеть сохраняет пары образцов и может восстановить образец, когда ассоциированный с ним образец предлагается ей в качестве подсказки. В этой сети два слоя элементов — по одному для каждого из образцов пары — и оба слоя соединяются двунаправленными связями (т.е. активность может передаваться по связям в обоих направлениях) (рис. 2).

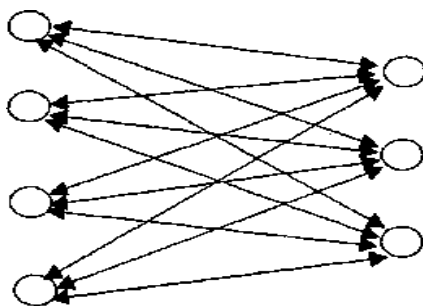


Рис. 2 Двухнаправленная ассоциативная память. Элементы слева представляют образцы размерности 4, а элементы справа - ассоциированные с ними образцы размерности 3

Рассмотрим только дискретную биполярную сеть ВАР, но можно рассмотреть и непрерывные значения. Чтобы сохранить образец  $s$  и ассоциируемый с ним образец  $t$ , рассматривается прямое произведение, определяющее весовые значения. Процедура точно такая же, как и в сети Хопфилда, но теперь матрица уже не обязана быть квадратной, а диагональные элементы не обнуляются. Весовой матрицей для одной пары будет матрица

$$W = s^T t.$$

Чтобы сохранить несколько пар, все соответствующие произведения, определяющие весовые значения, складываются, точно так же, как это делается для сети Хопфилда.

Процедура нахождения в памяти элемента подобна соответствующей процедуре сети Хопфилда. В следующем описании  $i$  обозначает один слой элементов, а  $j$  — ассоциированный слой элементов.

- Устанавливаются значения активности элементов слоя  $i$  в соответствии со значениями, задаваемыми входным образцом.
- Распространяется активность на слой  $j$ . Комбинированный ввод

элемента слоя  $j$  равен  $u_j = \sum_{i=1}^n s_i w_{ij}$

- Вычисляется новое состояние для каждого элемента слоя  $j$ :

$$t_j = f(u_j).$$

- Распространяется активность на слой  $i$ . Комбинированный ввод

элемента слоя  $i$  равен:  $s_i = f(u_i)$ .

•Это двустороннее распространение сигналов активности повторяется до тех пор, пока не будет достигнуто устойчивое состояние. Активность для каждого слоя определяется относительно некоторой пороговой величины  $\theta$ .

$$t_j = f(u_j) = \begin{cases} 1, \text{если } u_j > \theta_j \\ t_j, \text{если } u_j = \theta_j \\ -1, \text{если } u_j < \theta_j \end{cases} \quad s_i = f(u_i) = \begin{cases} 1, \text{если } u_i > \theta_i \\ s_i, \text{если } u_i = \theta_i \\ -1, \text{если } u_i < \theta_i \end{cases}$$

Все элементы сети сначала имеют нулевые значения активности. Обратите внимание на то, что распространение может начаться с любого уровня, так как  $s$  и  $t$  может использоваться для вызова  $t$ , и, наоборот,  $t$  может использоваться для вызова  $s$ .

#### ***Автоассоциативное обратное распространение ошибок***

Стандартная сеть с прямой связью и обратным распространением ошибок может быть обучена автоассоциативным методом выполнению задач типа сжатия изображений. Этот метод дает возможность сделать целевой образец таким же, как и учебный образец, так что сеть учится воспроизводить в выходном слое то, что подается ей на рассмотрение во входном слое. Базовая архитектура сети показана на рис. 4.

Сеть обучается стандартным образом, но целью обучения является ассоциация каждого учебного образца с самим собой. После успешного окончания обучения сеть может работать как два механизма: первый слой весов может обеспечивать сжатие образца, а второй слой весов может восстанавливать полный образец из его сжатого представления. Конечно, в результате реконструк-

ции могут быть и потери информации, поскольку сеть не

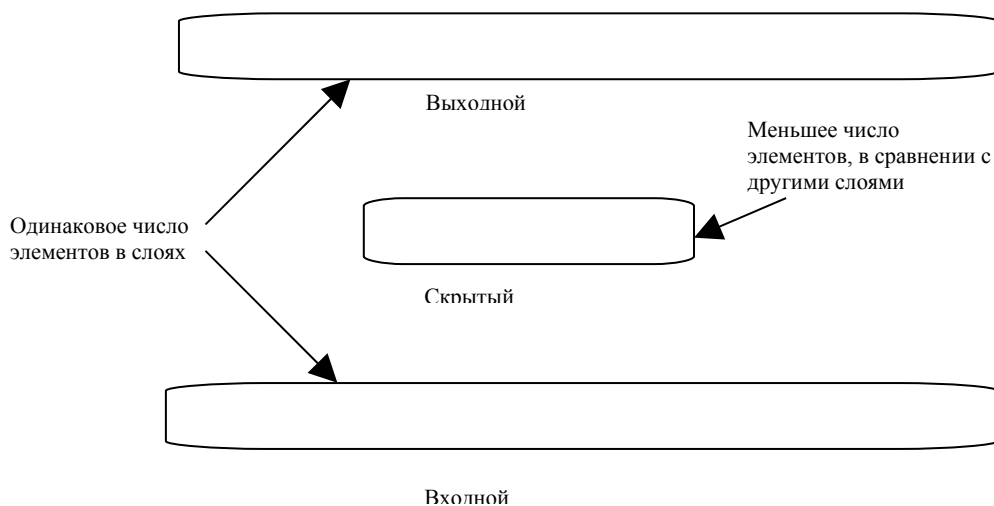


Рис. 4. Автоассоциативная сеть с прямой связью

будет в совершенстве создавать на выходном уровне все учебные экземпляры. Образец сжимается после его подачи на вход сети и распространения сигналов, по которым вычисляются значения активности скрытых элементов. Для сети с архитектурой типа 20-10-20 (20 входных элементов, 10 скрытых элементов и 20 выходных элементов) отношением сжатия будет 2-1, поскольку для данного входного вектора сжатый вектор задается значениями активности скрытых элементов. Чтобы восстановить входной вектор, сжатый вектор предъявляется скрытому слою, и активность распространяется на выходной слой.

Иногда может оказаться выгодным обработать данные с помощью автоассоциативной сети перед тем, как рассмотреть их с помощью другой сетевой модели (и даже перед тем, как выполнить их классификацию с помощью другой сети с прямой связью). Известно, например, что автоассоциативная сеть с прямой связью, имеющая один скрытый слой, вычисляет по сути главные компоненты. Анализ главных компонент является основным методом статистической обработки данных и используется для того, чтобы удалить избыточность данных и провести кластеризацию. В результате анализа главных



компонент происходит также декорреляция признаков векторов (признаки характеризуют положение элементов), что иногда оказывается полезным при обучении другой сети с прямой связью. Идея заключается в преобразовании учебных данных в некоторое сокращенное описание, где компонентами такого сокращенного описания выступают скрытые элементы.

Анализ главных компонент можно использовать для выполнения сжатия с потерями. Трансформация двумерных образцов в одномерные происходит следующим образом:

$$\begin{bmatrix} 1 & 1 \\ 3 & 3 \\ 4 & 4 \\ 8 & 8 \end{bmatrix} \begin{bmatrix} 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \end{bmatrix} = [1,414 \quad 4,243 \quad 5,657 \quad 11,314].$$

Обратное отображение для 1.414 задается с помощью следующего выражения:  $[1,414] \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = [1 \quad 1]$ .

Вывод: Ассоциативная сеть действует подобно памяти.

- Автоассоциация связывает образец с ним самим. Например, автоассоциативная память может использоваться для того, чтобы восстановить истинную версию сохраненного образца по искаженной версии этого образца.
- Гетероассоциация представляет собой связывание двух разных образцов. Один образец может использоваться в качестве подсказки, по которой из памяти извлекается другой образец.
- Когда автоассоциативное обучение происходит путем пропускания сигналов через узкий канал типа скрытого слоя автоассоциативной сети с прямой связью, происходит в некотором смысле сжатие данных. Сжатие может оказаться полезной формой предварительной обработки данных перед использованием их для обучения сети с другой архитектурой.
- Сеть Хопфилда может использоваться для автоассоциации, а двунаправленная ассоциативная память — для гетероассоциации.

- Весовые значения дискретной сети Хопфилда и двунаправленной ассоциативной памяти могут быть найдены с помощью простых матричных вычислений, поэтому такие сети не требуют длительного обучения.

- От числа элементов в ассоциативной сети зависит число образцов, которые могут быть сохранены.

Лекция № 8,9 **Тема: Градиентные алгоритмы обучения сети**

**Основные положения**

Задачу обучения нейронной сети будем рассматривать на данном этапе как требование минимизировать априори определенную целевую функцию  $E(w)$ . При таком подходе можно применять для обучения алгоритмы, которые в теории оптимизации считаются наиболее эффективными. К ним, без сомнения, относятся градиентные методы, чью основу составляет выявление градиента целевой функции. Они связаны с разложением целевой функции  $E(w)$  в ряд Тейлора в ближайшей окрестности точки имеющегося решения  $w$ . В случае целевой функции от многих переменных ( $w = [w_1, w_2, \dots, w_n]^T$ ) такое представление связывается с окрестностью ранее определенной точки (в частности, при старте алгоритма это исходная точка  $w_0$ ) в направлении  $p$ . Подобное разложение описывается универсальной формулой вида

$$E(w + p) = E(w) + [g(w)]^T p + \frac{1}{2} p^T H(w) p + \dots, \quad (1)$$

где  $g(w) = \nabla E = \left[ \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_n} \right]^T$  – это вектор градиента, а симметрич-

ная квадратная матрица

$$H(w) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1 \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \dots & \dots & \dots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \dots & \frac{\partial^2 E}{\partial w_n \partial w_n} \end{bmatrix}$$

является матрицей производных второго порядка, называемой *гессианом*.

В выражении (1)  $p$  играет роль направляющего вектора, зависящего от фактических значений вектора  $w$ . На практике чаще всего рассчитываются три первых члена ряда (1), а последующие просто игнорируются. При этом зависимость (1) может считаться квадратичным приближением целевой функции  $E(w)$  в ближайшей окрестности найденной точки  $w$  с точностью, равной локальной погрешности отсеченной части  $O(h^3)$ , где  $h = \|p\|$ . Для упрощения описания значения переменных, полученные в  $k$ -м цикле, будем записывать с нижним индексом  $k$ . Точкой решения  $w = w_k$  будем считать точку, в которой достигается минимум целевой функции  $E(w)$  и  $g(w_k) = 0$ , а гессиан  $H(w_k)$  является положительно определенным. При выполнении этих условий функция в любой точке, лежащей в окрестности  $w_k$ , имеет большее значение, чем в точке  $w_k$ , поэтому точка  $w_k$  является решением, соответствующим критерию минимизации целевой функции.

В процессе поиска минимального значения целевой функции направление поиска  $p$  и шаг  $h$  подбираются таким образом, чтобы для каждой очередной точки  $w_{k+1} = w_k + \eta_k p_k$  выполнялось условие  $E(w_{k+1}) < E(w_k)$ . Поиск минимума продолжается, пока норма градиента не упадет ниже априори заданного значения допустимой погрешности либо пока не будет превышено максимальное время вычислений (количество итераций).

Универсальный оптимизационный алгоритм обучения нейронной сети можно представить в следующем виде (будем считать, что начальное значение оптимизируемого вектора известно и составляет  $w_k = w_0$ ):

1. Проверка сходимости и оптимальности текущего решения  $w_k$ . Если точка  $w_k$  отвечает градиентным условиям остановки процесса – завершение вычислений. В противном случае перейти к п.2.

2. Определение вектора направления оптимизации  $p_k$  для точки  $w_k$ .
3. Выбор величины шага  $\eta_k$  в направлении  $p_k$ , при котором выполняется условие  $g(w_k)$ .
4. Определение нового решения  $w_{k+1} = w_k + \eta_k p_k$ , а также соответствующих ему значений  $E(w_k)$  и  $g(w_k)$ , а если требуется - то и  $H(w_k)$ , и возврат к п. 1.

### Алгоритм наискорейшего спуска

Если при разложении целевой функции  $E(w)$  в ряд Тейлора ограничиться ее линейным приближением, то мы получим алгоритм наискорейшего спуска. Для выполнения соотношения  $E(w_{k+1}) < E(w_k)$  достаточно подобрать  $g(w_k)^T p < 0$ . Условию уменьшения значения целевой функции отвечает выбор вектора направления

$$p_k = -g(w_k) \tag{2}$$

Именно выражением (2) определяется вектор направления  $p$  в методе наискорейшего спуска.

Ограничение слагаемым первого порядка при разложении функции в ряд Тейлора не позволяет использовать информацию о ее кривизне. Это обуславливает медленную сходимость метода (она остается линейной). Указанный недостаток, а также резкое замедление минимизации в ближайшей окрестности точки оптимального решения, когда градиент принимает очень малые значения, делают алгоритм наискорейшего спуска низкоэффективным. Тем не менее с учетом его простоты, невысоких требований к объему памяти и относительно небольшой вычислительной сложности именно этот метод в течение многих лет был и остается в настоящее время основным способом обучения многослойных сетей. Повысить его эффективность удастся путем модификации (как правило, эвристической) выражения, определяющего направление. Хорошие результаты приносит применение метода обучения с так называемым моментом. При этом подходе уточнение весов сети  $w_{k+1} = w_k + \Delta w_k$  производится с учетом модифицированной формулы определения значения  $\Delta w_k$

$$\Delta w_k = \eta_k p_k + \alpha (w_k - w_{k-1}), \quad (3)$$

где  $\alpha$  – это коэффициент момента, принимающий значения в интервале  $[0, 1]$ . Первое слагаемое этого выражения соответствует обычному обучению по методу наискорейшего спуска, тогда как второе учитывает последнее изменение весов и не зависит от фактического значения градиента. Чем больше значение коэффициента  $\alpha$ , тем большее значение оказывает показатель момента на подбор весов. Это влияние существенно возрастает на плоских участках целевой функции, а также вблизи локального минимума, где значение градиента близко к нулю.

На плоских участках целевой функции приращение весов (при постоянном значении коэффициента обучения  $\eta_k = \eta$ ) остается приблизительно одним и тем же. Это означает, что  $\Delta w_k = \eta p_k + \alpha \Delta w_k$ , поэтому эффективное приращение значений весов можно описать отношением

$$\Delta w_k = \frac{\eta}{1 - \alpha} p_k \quad (4)$$

При значении  $\alpha = 0,9$  это соответствует 10-кратному увеличению эффективного значения коэффициента обучения и, следовательно, также 10-кратному ускорению процесса обучения.

Вблизи локального минимума показатель момента, не связанный с градиентом, может вызвать слишком большое изменение весов, приводящее к увеличению значения целевой функции и к выходу из "зоны притяжения" этого минимума. При малых значениях градиента показатель момента начинает доминировать в выражении (3), что приводит к такому приращению весов  $\Delta w_k$ , которое соответствует увеличению значения целевой функции, позволяющему выйти из зоны локального минимума. Однако показатель момента не должен полностью доминировать на протяжении всего процесса обучения, поскольку это привело бы к нестабильности алгоритма. Для предотвращения такого избыточного доминирования значение целевой функции  $E$  контролируется так, чтобы допускать его увеличение только в ограничен-

ных пределах, например не более 4%. При таком подходе, если на очередных ( $k$ -м и  $(k+1)$ -м) шагах итерации выполняется условие  $E(k+1) < 1,04E(k)$ , то изменения игнорируются и считается, что  $(w_k - w_{k-1}) = 0$ . При этом показатель градиента начинает доминировать над показателем момента и процесс развивается в направлении минимизации, заданном вектором градиента. Следует подчеркнуть, что подбор величины коэффициента момента является непростым делом и требует проведения большого количества экспериментов, имеющих целью выбрать такое значение, которое наилучшим образом отражало бы специфику решаемой проблемы.

### Алгоритм переменной метрики

В методе переменной метрики используется квадратичное приближение функции  $E(w)$  в окрестности полученного решения  $\Delta w_k$ . Если в формуле (1) ограничиться тремя первыми слагаемыми, то получим:

$$E(w_k + p_k) \cong E(w_k) + g(w_k)^T p_k + \frac{1}{2} p_k^T H(w_k) p_k + O(h^3). \quad (5)$$

Для достижения минимума функции (5) требуется, чтобы

$$\frac{dE(w_k + p_k)}{dp_k} = 0. \quad \text{При выполнении соответствующего дифференцирования}$$

можно получить условие оптимальности в виде

$$g(w_k) + H(w_k)p_k = 0.$$

Элементарное преобразование этого выражения дает очевидное решение:

$$p_k = -[H(w_k)]^{-1} g(w_k) \quad (6)$$

Формула (6) однозначно указывает направление  $p_k$ , которое гарантирует достижение минимального для данного шага значения целевой функции. Из него следует, что для определения этого направления необходимо в каждом цикле вычислять значение градиента  $g$  и гессиана  $H$  в точке известного (последнего) решения  $w_k$ .

Формула (6) представляющая собой основу ньютоновского алгоритма оптимизации, является чисто теоретическим выражением, поскольку ее применение требует положительной определенности гессиана на каждом шаге, что в общем случае практически неосуществимо. По этой причине в имеющихся реализациях алгоритма, как правило, вместо точно определенного гессиана  $H(w_k)$  используется его приближение  $G(w_k)$ . Одним из наиболее популярных считается метод переменной метрики. В соответствии с этим методом на каждом шаге гессиан или обратная ему величина, полученная на предыдущем шаге, модифицируется на величину некоторой поправки. Если прирост вектора  $w_k$  и градиента  $g$  на двух последовательных шагах итерации обозначить соответственно  $s_k$  и  $r_k$ , т.е.  $s_k = w_k - w_{k-1}$  и  $r_k = g(w_k) - g(w_{k-1})$ , а матрицу, обратную приближению гессиана  $V_k = [G(w_k)]^{-1}$ ,  $V_{k-1} = [G(w_{k-1})]^{-1}$ , обозначить  $V$ , то в соответствии с очень эффективной формулой Бroyдена-Флетчера-Гольдфарба-Шенно (BFGS) процесс уточнения значения матрицы  $V$  можно описать рекуррентной зависимостью:

$$V_k = V_{k-1} + \left[ 1 + \frac{r_k^T V_{k-1} r_k}{s_k^T r_k} \right] \frac{s_k s_k^T}{s_k^T r_k} - \frac{s_k r_k^T V_{k-1} r_k s_k^T}{s_k^T r_k}. \quad (7)$$

В другом известном алгоритме Девидона-Флетчера-Пауэлла (DFP) значение гессиана уточняется согласно выражению

$$V_k = V_{k-1} + \frac{s_k s_k^T}{s_k^T r_k} - \frac{V_{k-1} r_k r_k^T V_{k-1}}{r_k^T V_{k-1} r_k}. \quad (8)$$

В качестве начального значения обычно принимается  $V_0 = 1$ , а первая итерация проводится в соответствии с алгоритмом наискорейшего спуска. При начальном значении  $V_0 = 1$  и при использовании направленной минимизации на каждом шаге оптимизации можно обеспечить положительную определенность аппроксимированной матрицы гессиана. Направленная минимизация необходима при реализации как стратегии BFGS, так и DFP, причем в соответствии с проведенными тестами метод BFGS менее чувствителен к различным погрешностям

вычислительного процесса. По этой причине, несмотря на несколько большую вычислительную сложность, метод BFGS применяется чаще, чем DFP.

Метод переменной метрики характеризуется более быстрой сходимостью, чем метод наискорейшего спуска. Кроме того, факт положительной определенности гессиана на каждом шаге итерации придает уверенность в том, что выполнение условия  $g(w_k) = 0$  действительно гарантирует решение проблемы оптимизации. Именно этот метод считается в настоящее время одним из наиболее эффективных способов оптимизации функции нескольких переменных. Его недостаток состоит в относительно большой вычислительной сложности (связанной с необходимостью расчета в каждом цикле  $n^2$  элементов гессиана), а также в использовании значительных объемов памяти для хранения элементов гессиана, что в случае оптимизации функции с большим количеством переменных может стать серьезной проблемой. По этой причине метод переменной метрики применяется для не очень больших сетей. В частности, с использованием персонального компьютера была доказана его эффективность для сети, содержащей не более тысячи взвешенных связей.

### Алгоритм Левенберга-Марквардта

Другим приложением ньютоновской стратегии оптимизации является алгоритм Левенберга-Марквардта. При его использовании точное значение гессиана  $H(w)$  в формуле (6) заменяется аппроксимированным значением  $G(w)$ , которое рассчитывается на основе содержащейся в градиенте информации с учетом некоторого регуляризационного фактора.

Для описания этого метода представим целевую функцию в виде, отвечающем существованию единственной обучающей выборки,

$$E(w) = \frac{1}{2} \sum_{i=1}^M [e_i(w)]^2, \quad (9)$$

где  $e_i = [y_i(w) - d_i]$ . При использовании обозначений



$$e(w) = \begin{bmatrix} e_1(w) \\ e_2(w) \\ \dots \\ e_M(w) \end{bmatrix}, \quad J(w) = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \dots & \frac{\partial e_1}{\partial w_n} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \dots & \frac{\partial e_2}{\partial w_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_M}{\partial w_1} & \frac{\partial e_M}{\partial w_2} & \dots & \frac{\partial e_M}{\partial w_n} \end{bmatrix}, \quad (10)$$

вектор градиента и аппроксимированная матрица гессиана, соответствующие целевой функции (9), определяются в виде

$$g(w) = [J(w)]^T e(w), \quad (11)$$

$$G(w) = [J(w)]^T J(w) + R(w), \quad (12)$$

где  $R(w)$  обозначены компоненты гессиана  $H(w)$ , содержащие высшие производные относительно  $w$ . Сущность подхода Левенберга-Марквардта состоит в аппроксимации  $R(w)$  с помощью регуляризационного фактора  $\nu$  1, в котором переменная  $\nu$ , называемая параметром Левенберга-Марквардта, является скалярной величиной, изменяющейся в процессе оптимизации. Таким образом, аппроксимированная матрица гессиана на  $k$ -м шаге алгоритма приобретает вид:

$$G(w_k) = [J(w_k)]^T J(w_k) + \nu_k 1. \quad (13)$$

В начале процесса обучения, когда фактическое значение  $w_k$  еще далеко от искомого решения (велико значение вектора погрешности  $e$ ), используется значение параметра  $\nu_k$ , намного превышающее собственное значение матрицы  $[J(w_k)]^T J(w_k)$ . В таком случае гессиан фактически подменяется регуляризационным фактором:

$$G(w_k) \cong \nu_k 1, \quad (14)$$

а направление минимизации выбирается по методу наискорейшего спуска:

$$p_k = - \frac{g(w_k)}{\nu_k}. \quad (15)$$

По мере уменьшения погрешности и приближения к искомому решению величина параметра  $v_k$  понижается и первое слагаемое в формуле (12) начинает играть все более важную роль.

На эффективность алгоритма влияет грамотный подбор величины  $v_k$ . Слишком большое начальное значение  $v_k$  по мере прогресса оптимизации должно уменьшаться вплоть до нуля при достижении фактического решения, близкого к искомому. Известны различные способы подбора этого значения, но мы ограничимся описанием только одной оригинальной методики, предложенной Д. Марквардтом. Пусть значения целевой функции на  $k$ -м и  $(k-1)$ -м шагах итерации обозначаются соответственно  $E_k$  и  $E_{k-1}$ , а значения параметра  $v$  на этих же шагах —  $v_k$  и  $v_{k-1}$ . Коэффициент уменьшения значения  $v$  обозначим  $r$ , причем  $r > 1$ . В соответствии с классическим алгоритмом Левенберга-Марквардта значение  $v$  изменяется по следующей схеме:

- если  $E\left(\frac{v_{k-1}}{r}\right) \leq E_k$ , то принять  $v_k = \frac{v_{k-1}}{r}$ ;
- если  $E\left(\frac{v_{k-1}}{r}\right) > E_k$  и  $E(v_{k-1}) < E_k$ , то принять  $v_k = v_{k-1}$ ;
- если  $E\left(\frac{v_{k-1}}{r}\right) > E_k$  и  $E(v_{k-1}) > E_k$ , то увеличить последовательно  $m$  раз значение  $v$  до достижения  $E(v_{k-1}r^m) \leq E_k$ , одновременно принимая  $v_k = v_{k-1}r^m$ .

Такая процедура изменения значения  $v$  выполняется до момента, в котором так называемый коэффициент верности отображения  $q$ , рассчитываемый по формуле

$$q = \frac{E_k - E_{k-1}}{[\Delta w_k]^T g_k + 0,5[\Delta w_k]^T G_k \Delta w_k}, \quad (16)$$

достигнет значения, близкого к единице. При этом квадратичная аппроксимация целевой функции имеет высокую степень совпадения с истинными значениями, что свидетельствует о близости оптимального решения. В такой ситуации регуляризационный фактор  $v_k$  в формуле (13) может быть опущен ( $v_k = 0$ ), процесс

определения гессиана сводится к непосредственной аппроксимации первого порядка, а алгоритм Левенберга-Марквардта превращается в алгоритм Гаусса-Ньютона, характеризующийся квадратичной сходимостью к оптимальному решению.

### Алгоритм сопряженных градиентов

В этом методе при выборе направления минимизации не используется информация о гессиане. Направление поиска  $p_k$  выбирается таким образом, чтобы оно было ортогональным и сопряженным ко всем предыдущим направлениям  $p_0, p_1, \dots, p_{k-1}$ . Множество векторов  $p_i, i = 0, 1, \dots, k$ , будет взаимно сопряженным относительно матрицы  $G$ , если

$$p_i^T G p_j = 0, \quad i \neq j. \quad (17)$$

Вектор  $p_k$ , удовлетворяющий заданным выше условиям, имеет вид:

$$p_k = -g_k + \beta_{k-1} p_{k-1} \quad (18)$$

где  $g_k = g(w_k)$  обозначает фактическое значение вектора градиента.

Из формулы (18) следует, что новое направление минимизации зависит только от значения градиента в точке решения  $w_k$  и от предыдущего направления поиска  $p_{k-1}$ , умноженного на коэффициент сопряжения  $\beta_{k-1}$ . Этот коэффициент играет очень важную роль, аккумулируя в себе информацию о предыдущих направлениях поиска. Существуют различные правила расчета его значения. Наиболее известны среди них

$$\beta_{k-1} = \frac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}} \quad (19)$$

$$\text{и} \quad \beta_{k-1} = \frac{g_k^T (g_k - g_{k-1})}{-p_{k-1}^T g_{k-1}} \quad (20)$$

Ввиду накопления погрешностей округления в последовательных циклах вычислений практическое применение метода сопряженных градиентов связано с постепенной утратой свойства ортогональности между векторами направлений минимизации. По этой причине после выполнения  $n$  итераций (значение  $n$  рассчитывается как функция от количества переменных, подлежащих оптимизации)

производится рестарт процедуры, на первом шаге которой направление минимизации из точки полученного решения выбирается по алгоритму наискорейшего спуска. Метод сопряженных градиентов имеет сходимость, близкую к линейной, и он менее эффективен, чем метод переменной метрики, однако заметно быстрее метода наискорейшего спуска. Он широко применяется как единственно эффективный алгоритм оптимизации при весьма значительном количестве переменных, которое может достигать нескольких десятков тысяч. Благодаря невысоким требованиям к памяти и относительно низкой вычислительной сложности метод сопряженных градиентов позволяет успешно решать очень серьезные оптимизационные задачи.

### Лекция № 9 Подбор коэффициента обучения

Рассмотренные алгоритмы позволяют определить только направление, в котором уменьшается целевая функция, но не говорят ничего о величине шага, при котором эта функция может получить минимальное значение. После выбора правильного направления  $p_k$  следует определить на нем новую точку решения  $w_{k+1}$ , в которой будет выполняться условие  $E(w_{k+1}) < E(w_k)$ . Необходимо подобрать такое значение  $\eta_k$ , чтобы новое решение  $w_{k+1} = w_k + \eta_k p_k$  лежало как можно ближе к минимуму функции  $E(w)$  в направлении  $p_k$ . Грамотный подбор коэффициента  $\eta_k$  оказывает огромное влияние на сходимость алгоритма оптимизации к минимуму целевой функции. Чем сильнее величина  $\eta_k$  отличается от значения, при котором  $E(w)$  достигает минимума в выбранном направлении  $p_k$ , тем большее количество итераций потребуется для поиска оптимального решения. Слишком малое значение  $\eta$  не позволяет минимизировать целевую функцию за один шаг и вызывает необходимость повторно двигаться в том же направлении. Слишком большой шаг приводит к "перепрыгиванию" через минимум функции и фактически заставляет возвращаться к нему.

Существуют различные способы подбора значения  $\eta$ , называемого в теории нейронных сетей *коэффициентом обучения*. Простейший из них (относительно редко применяемый в настоящее время, главным образом для обучения в режи-

ме "онлайн") основан на фиксации постоянного значения  $\eta$  на весь период оптимизации. Этот способ практически используется только совместно с методом наискорейшего спуска. Он имеет низкую эффективность, поскольку значение коэффициента обучения никак не зависит от вектора фактического градиента и, следовательно, от направления  $p$  на данной итерации. Величина  $\eta$  подбирается, как правило, отдельно для каждого слоя сети с использованием различных эмпирических зависимостей. Один из подходов состоит в определении минимального значения коэффициента  $\eta$  для каждого слоя по формуле

$$\eta \leq \min\left(\frac{1}{n_i}\right), \quad (1)$$

где  $n_i$  обозначает количество входов  $i$ -го нейрона в слое.

Другой более эффективный метод основан на адаптивном подборе коэффициента  $\eta$  с учетом фактической динамики величины целевой функции в результате обучения. В соответствии с этим методом стратегия изменения значения  $\eta$  определяется путем сравнения суммарной погрешности  $\varepsilon$   $i$ -й итерации с ее предыдущим значением, причем  $\varepsilon$  рассчитывается по формуле

$$\varepsilon = \sqrt{\sum_{j=1}^M (y_j - d_j)^2} \quad (2)$$

Для ускорения процесса обучения следует стремиться к непрерывному увеличению  $\eta$  при одновременном контроле прироста погрешности  $\varepsilon$  по сравнению с ее значением на предыдущем шаге. Незначительный рост этой погрешности считается допустимым.

Если погрешности на  $(i-1)$  и  $i$ -й итерациях обозначить соответственно  $\varepsilon_{i-1}$  и  $\varepsilon_i$ , а коэффициенты обучения на этих же итерациях -  $\eta_{i-1}$  и  $\eta_i$ , то в случае  $\varepsilon_i > k_w \varepsilon_{i-1}$  ( $k_w$  - коэффициент допустимого прироста погрешности) значение  $\eta$  должно уменьшаться в соответствии с формулой

$$\eta_{i+1} = \eta_i P_d, \quad (3)$$

где  $P_d$  - коэффициент уменьшения  $\eta$ . В противном случае, когда  $\varepsilon_i \leq k_w \varepsilon_{i-1}$ , принимается

$$\eta_{i+1} = \eta_i p_i \quad (4)$$

где  $p_i$  - коэффициент увеличения  $\eta$ . Несмотря на некоторое возрастание объема вычислений (необходимых для дополнительного расчета значений  $\varepsilon$ ), возможно существенное ускорение процесса обучения. Например, реализация представленной стратегии в программе MATLAB со значениями  $k_w = 1,41$ ,  $p_d = 0,7$ ,  $p_i = 1,05$  позволила в несколько раз ускорить обучение при решении проблемы аппроксимации нелинейных функций.

Интересно проследить характер изменения коэффициента  $\eta$  в процессе обучения. Как правило, на начальных этапах доминирует тенденция к его увеличению, однако при достижении некоторого квазистационарного состояния величина  $\eta$  постепенно уменьшается, но не монотонно, а циклически возрастая и понижаясь в следующих друг за другом циклах.

Однако необходимо подчеркнуть, что адаптивный метод подбора  $\eta$  сильно зависит от вида целевой функции и значений коэффициентов  $k_w$ ,  $p_d$  и  $p_i$ . Значения, оптимальные для функции одного вида, могут замедлять процесс обучения при использовании другой функции. Поэтому при практической реализации этого метода следует обращать внимание на механизмы контроля и управления значениями коэффициентов, подбирая их в соответствии со спецификой решаемой задачи.

Наиболее эффективный, хотя и наиболее сложный, метод подбора коэффициента обучения связан с направленной минимизацией целевой функции в выбранном заранее направлении  $p_k$ . Необходимо так подобрать скалярное значение  $\eta_k$ , чтобы новое решение  $w_{k+1} = w_k + \eta_k p_k$  соответствовало минимуму целевой функции в данном направлении  $p_k$ . В действительности получаемое решение  $w_{k+1}$  только с определенным приближением может считаться настоящим минимумом. Это результат компромисса между объемом вычислений и влиянием величины  $\eta_k$  на сходимость алгоритма.

Среди наиболее популярных способов направленной минимизации можно выделить *безградиентные* и *градиентные методы*. В *безградиентных методах*

используется только информация о значениях целевой функции, а ее минимум достигается в процессе последовательного уменьшения диапазона значений вектора  $w$ . Примерами могут служить методы деления пополам, золотого сечения либо метод Фибоначчи, различающиеся способом декомпозиции получаемых поддиапазонов.

Заслуживает внимания метод аппроксимации целевой функции  $E\{w\}$  в предварительно выбранном направлении  $P_k$  с последующим расчетом минимума, получаемого таким образом, функции одной переменной  $\eta$ . Выберем для аппроксимации многочлен второго порядка вида

$$E(w) \rightarrow P_2(\eta) = a_2\eta^2 + a_1\eta + a_0, \quad (5)$$

где  $a_2, a_1, a_0$  обозначены коэффициенты, определяемые в каждом цикле оптимизации. Выражение (5) - это многочлен  $P_2$  одной скалярной переменной  $\eta$ . Если для расчета входящих в  $P_2$  коэффициентов используются три произвольные точки  $w_1, w_2$  и  $w_3$ , лежащие в направлении  $P_k$ , т.е.  $w_1 = w + \eta_1 P_k$ ,  $w_2 = w + \eta_2 P_k$ ,  $w_3 = w + \eta_3 P_k$  (в этом выражении за  $w$  обозначено предыдущее решение), а соответствующие этим точкам значения целевой функции  $E\{w\}$  обозначены  $E_1 = E(w_1)$ ,  $E_2 = E(w_2)$ ,  $E_3 = E(w_3)$ , то

$$P_2(\eta_1) = E_1, P_2(\eta_2) = E_2, P_2(\eta_3) = E_3. \quad (6)$$

Коэффициенты  $a_2, a_1, a_0$  многочлена  $P_2$  рассчитываются в соответствии с системой линейных уравнений, описываемых в (6). Для определения ми-

нимума этого многочлена его производная  $\frac{dP_2}{d\eta} = 2a_2\eta + a_1$  приравняется к

нулю, что позволяет получить значение  $\eta$  в виде  $\eta_{\min} = \frac{-a_1}{2a_2}$ . После подстановки

выражений для  $E_1, E_2$  и  $E_3$  в формулу расчета  $\eta_{\min}$  получаем:

$$\eta_{\min} = \eta_2 - \frac{1}{2} \frac{(\eta_2 - \eta_1)^2(E_2 - E_3) - (\eta_2 - \eta_3)^2(E_2 - E_1)}{(\eta_2 - \eta_1)(E_2 - E_3) - (\eta_2 - \eta_3)(E_2 - E_1)} \quad (7)$$

Однако лучшим решением считается применение градиентных методов, в которых, кроме значения функции, учитывается также и ее производная вдоль направляющего вектора  $p_k$ . Они позволяют значительно ускорить достижение минимума, поскольку используют информацию о направлении уменьшения величины целевой функции. В такой ситуации применяется, как правило, аппроксимирующий многочлен третьего порядка

$$P_3(\eta) = a_3\eta^3 + a_2\eta^2 + a_1\eta + a_0. \quad (8)$$

Значения четырех коэффициентов  $a_i$  этого многочлена можно получить исходя из информации о величине функции и ее производной всего лишь в двух точках. Если приравнять к нулю производную многочлена относительно  $\eta$ , то можно получить формулу для расчета  $\eta_{\min}$  в виде

$$\eta_{\min} = \frac{-a_2 + \sqrt{a_2^2 - 3a_2a_1}}{3a_3}. \quad (9)$$

### Подбор оптимальной архитектуры сети

Одно из важнейших свойств нейронной сети – это способность к обобщению полученных знаний. Сеть, натренированная на некотором множестве обучающих выборок, генерирует ожидаемые результаты при подаче на ее вход данных, относящихся к тому же множеству, но не участвовавших непосредственно в процессе обучения. Разделение данных на обучающее и тестовое подмножества представлено на рис. 1. Множество данных, на котором считается истинным некоторое правило  $R$ , разбито на подмножества  $L$  и  $G$ , при этом в составе  $L$ , в свою очередь, можно выделить определенное подмножество контрольных данных  $V$ , используемых для верификации степени обучения сети. Обучение проводится на данных, составляющих подмножество  $L$ . Способность отображения сетью

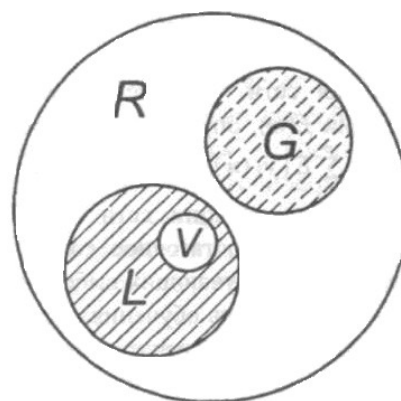


Рис. 1. Иллюстрация разделения данных.



элементов  $L$  может считаться показателем степени накопления обучающих данных, тогда как способность опознавания данных, входящих во множество  $G$  и не использованных для обучения, характеризует ее возможности обобщения (генерализации) знаний. Данные, входящие и в  $L$ , и в  $G$ , должны быть типичными элементами множества  $R$ . В обучающем подмножестве не должно быть уникальных данных, свойства которых отличаются от ожидаемых (типичных) значений.

Феномен обобщения возникает вследствие большого количества комбинаций входных данных, которые могут кодироваться в сети с  $N$  входами. Если в качестве простого примера рассмотреть однослойную сеть с одним выходным нейроном, то для нее может быть составлено  $2^N$  входных выборок. Каждой выборке может соответствовать единичное или нулевое состояние выходного нейрона. Таким образом, общее количество различаемых сигналов составит  $2^N$ . Если для обучения сети используются  $p$  из общего числа  $2^N$  входных выборок, то оставшиеся незадействованными  $(2^N - p)$  допустимых комбинаций характеризуют потенциально возможный уровень обобщения знаний.

Подбор весов сети в процессе обучения имеет целью найти такую комбинацию их значений, которая наилучшим образом воспроизводила бы последовательность ожидаемых обучающих пар  $(x_i, d_i)$ . При этом наблюдается тесная связь между количеством весов сети (числом степеней свободы) и количеством обучающих выборок. Если бы целью обучения было только запоминание обучающих выборок, их количество могло быть равным числу весов. В таком случае каждый вес соответствовал бы единственной обучающей паре. К сожалению, такая сеть не будет обладать свойством обобщения и сможет только восстанавливать данные. Для обретения способности обобщать информацию сеть должна тренироваться на избыточном множестве данных, поскольку тогда веса будут адаптироваться не к уникальным выборкам, а к их статистически усредненным совокупностям. Следовательно, для усиления способности к обобщению необходимо не только оптимизировать структуру сети в

направлении ее минимизации, но и оперировать достаточно большим объемом обучающих данных.

Обратим внимание на определенную непоследовательность процесса обучения сети. Собственно обучение ведется путем минимизации целевой функции  $E\{w\}$ , определяемой только на обучающем подмножестве  $L$ , при этом

$$E(w) = \sum_{k=1}^p E(y_k(w), d_k), \text{ где } p \text{ обозначено количество обучающих пар } (x_k, d_k), \text{ а } y_k$$

- вектор реакции сети на возбуждение  $x_k$ . Минимизация этой функции обеспечивает достаточное соответствие выходных сигналов сети ожидаемым значениям из обучающих выборок.

Истинная цель обучения состоит в таком подборе архитектуры и параметров сети, которые обеспечат минимальную погрешность распознавания тестового подмножества данных, не участвовавших в обучении. Эту погрешность будем называть погрешностью обобщения  $E_G(w)$ . Со статистической точки зрения погрешность обобщения зависит от уровня погрешности обучения  $E_L(w)$  и от доверительного интервала  $\varepsilon$ . Она характеризуется отношением

$$E_G(w) \leq E_L(w) + \varepsilon \left( \frac{p}{h}, E_L \right) \quad (1)$$

Значение  $\varepsilon$  функционально зависит от уровня погрешности обучения  $E_L(w)$  и от отношения количества обучающих выборок  $p$  к фактическому значению  $\eta$  параметра, называемого мерой Вапника-Червоненкиса и обозначаемого  $VCdim$ . Мера  $VCdim$  отражает уровень сложности нейронной сети и тесно связана с количеством содержащихся в ней весов. Значение  $\varepsilon$  уменьшается по мере возрастания отношения количества обучающих выборок к уровню сложности сети.

По этой причине обязательным условием выработки хороших способностей к обобщению считается грамотное определение меры Вапника-Червоненкиса для сети заданной структуры. Метод точного определения этой меры не известен, о нем можно лишь сказать, что ее значение функционально зависит от количества синаптических весов, связывающих нейроны между собой. Чем больше количе-

ство различных весов, тем больше сложность сети и соответственно значение меры  $VCdim$ . Будем определять верхнюю и нижнюю границы этой меры в виде

$$2 \left\lfloor \frac{K}{2} \right\rfloor N \leq VCdim \leq 2N_w(1 + \lg N_n), \quad (2)$$

где  $\lfloor \cdot \rfloor$  обозначена целая часть числа,  $N$  - размерность входного вектора,  $K$  - количество нейронов скрытого слоя,  $N_w$  - общее количество весов сети, а  $N_n$  - общее количество нейронов сети.

Из выражения (2) следует, что нижняя граница диапазона приблизительно равна количеству весов, связывающих входной и скрытый слои, тогда как верхняя граница превышает двукратное суммарное количество всех весов сети. В связи с невозможностью точного определения меры  $VCdim$  в качестве ее приближенного значения используется общее количество весов нейронной сети.

Таким образом, на погрешность обобщения оказывает влияние отношение количества обучающих выборок к количеству весов сети. Небольшой объем обучающего подмножества при фиксированном количестве весов вызывает хорошую адаптацию сети к его элементам, однако не усиливает способности к обобщению, так как в процессе обучения наблюдается относительное превышение числа подбираемых параметров (весов) над количеством пар фактических и ожидаемых выходных сигналов сети. Эти параметры адаптируются с чрезмерной (а вследствие превышения числа параметров над объемом обучающего множества - и неконтролируемой) точностью к значениям конкретных выборок, а не к диапазонам, которые эти выборки должны представлять. Фактически задача аппроксимации подменяется в этом случае задачей приближенной интерполяции. В результате всякого рода нерегулярности обучающих данных и измерительные шумы могут восприниматься как существенные свойства процесса. Функция, воспроизводимая в точках обучения, будет хорошо восстанавливаться только при соответствующих этим точкам значениях. Даже минимальное отклонение от этих точек вызовет значительное увеличение погрешности, что будет

восприниматься как ошибочное обобщение. По результатам разнообразных численных экспериментов установлено, что высокие показатели обобщения достигаются в случае, когда количество обучающих выборок в несколько раз превышает меру  $VCdim$ .

Решение по выбору окончательной схемы сети может быть принято только после полноценного обучения (с уменьшением погрешности до уровня, признаваемого удовлетворительным) различных вариантов ее структуры. Однако нет никакой уверенности в том, что этот выбор будет оптимальным, поскольку тренируемые сети могут отличаться различной чувствительностью к подбору начальных значений весов и параметров обучения. По этой причине базу для редукции сети составляют алгоритмы отсеечения взвешенных связей либо исключения нейронов в процессе обучения или после его завершения.

Как правило, методы непосредственного отсеечения связей, основанные на временном присвоении им нулевых значений, с принятием решения о возобновлении их обучения по результатам наблюдаемых изменений величины целевой функции (если это изменение слишком велико, следует восстановить отсеченную связь), оказываются неприменимыми из-за слишком высокой вычислительной сложности. Большинство применяемых в настоящее время алгоритмов редукции сети можно разбить на две категории. Методы первой группы исследуют чувствительность целевой функции к удалению веса или нейрона. С их помощью устраняются веса с наименее заметным влиянием, оказывающие минимальное воздействие на величину целевой функции, и процесс обучения продолжается уже на редуцированной сети.

Методы второй группы связаны с модификацией целевой функции, в которую вводятся компоненты, штрафующие за неэффективную структуру сети. Чаще всего это бывают элементы, усиливающие малые значения амплитуды весов. Такой способ менее эффективен по сравнению с методами первой группы, поскольку малые значения весов не обязательно ослабляют их влияние на функционирование сети.

Принципиально иной подход состоит в начале обучения при минимальном (обычно нулевом) количестве скрытых нейронов и последовательном их добавлении вплоть до достижения требуемого уровня натренированности сети на исходном множестве обучающих выборок. Добавление нейронов, как правило, производится по результатам оценивания способности сети к обобщению после определенного количества циклов обучения. В частности, именно такой прием реализован в алгоритме каскадной корреляции Фальмана.

При обсуждении способности сети к обобщению невозможно обойти вниманием влияние на ее уровень длительности обучения. Численные эксперименты показали, что погрешность обучения при увеличении количества итераций монотонно уменьшается, тогда как погрешность обобщения снижается только до определенного момента, после чего начинает расти.

Излишние веса адаптируются к любым нерегулярностям обучающих данных, воспринимая их в качестве важных характеристик. Как следствие, на этапе тестирования они становятся причиной возникновения значительных погрешностей воспроизведения.

Для предупреждения переобучения в обучающем множестве выделяется область контрольных данных (подмножество  $V$  на рис. 1), которые в процессе обучения применяются для оперативной проверки фактически набранного уровня обобщения.

Обучение прекращается, когда погрешность обобщения на этом подмножестве достигнет минимального значения (или начнет возрастать).

## **7. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ**

## ЛАБОРАТОРНЫХ РАБОТ

### Лабораторная работа № 1

#### Тема: Создание однонаправленной сети

Цель занятия – продемонстрировать основные этапы реализации нейронно-сетевого подхода для решения конкретной задачи. Можно выделить 4 основных этапа:

1. Подготовка данных для тренировки сети.
2. Создание сети.
3. Обучение сети.
4. Тестирование сети.
5. Моделирование сети. (Использование сети для решения поставленной задачи.)

В качестве примера рассмотрим следующую задачу:

Задан массив, состоящий из нескольких значений функции  $y = Ce^{\frac{-(x-A)^2}{S}}$  (S>0)

на интервале (0,1). Создать нейронную сеть такую, что при вводе этих значений на входы сети на выходах получались бы значения параметров C, A и S.

#### 1. Подготовка данных для обучения сети

В первую очередь необходимо определиться с размерностью входного массива.

Выберем количество значений функции равным N=21, т.е. в качестве входных векторов массива используем значения функции y в точках x=0.05; ...1.0. Для обучения сети необходимо сформировать массив входных векторов для различных наборов параметров C, A и S. Каждый набор этих параметров является вектором-эталоном для соответствующего входного вектора.

Для подготовки входного и эталонного массивов воспользуемся следующим алгоритмом. Выбираем случайным образом значения компонент вектора – эталона C, A S и вычисляем компоненты соответствующего входного вектора. Повторяем эту процедуру M раз и получаем массив входных векторов в виде матрицы размерностью NxM и массив векторов – эталонов в виде матрицы раз-

мерностью в нашем случае  $3 \times M$ . Полученные массивы мы можем использовать для обучения сети.

Прежде чем приступить к формированию обучающих массивов необходимо определиться с некоторыми свойствами массивов.

1. Диапазон изменения параметров  $C, A, S$ . Выберем диапазоны изменения параметров  $C, A, S$  равными  $(0.1, 1)$ . Значения, близкие к 0 и сам 0 исключим в связи с тем, что функция не определена при  $S=0$ . Второе ограничение связано с тем, что при использовании типичных передаточных функций желательно, чтобы компоненты входных и выходных векторов не выходили за пределы диапазона  $(-1, 1)$ . В дальнейшем мы познакомимся с методами нормировки, которые позволяют обойти это ограничение.
2. Количество входных и эталонных векторов выберем равным  $M=100$ . Этого достаточно для обучения а процесс обучения не займет много времени.

Тестовые массивы и эталоны подготовим с помощью программы *mas1*:

% формирование входных массивов (входной массив P) и (эталонны T)

```
P=zeros(100,21);
T=zeros(3,100);
x=0:5.e-2:1;
for i=1:100
c=0.9*rand+0.1;
a=0.9*rand+0.1;
s=0.9*rand+0.1;
T(1,i)=c;
T(2,i)=a;
T(3,i)=s;
P(i,:)=c*exp(-((x-a).^2/s));
end;
P=P';
```

С помощью этой программы формируется матрица P из  $M=100$  входных векторов-столбцов, каждый из которых сформирован из 21 точки исходной

функции со случайно выбранными значениями параметров  $C, A, S$ , и матрица  $T$  эталонов из 100 эталонных векторов-столбцов, каждый из которых сформирован из 3 соответствующих эталонных значений. Матрицы  $P$  и  $T$  будут использованы при обучении сети. Следует отметить, что при каждом новом запуске этой программы будут формироваться массивы с новыми значениями компонентов векторов, как входных, так и эталонных.

## 2. Создание сети

Вообще, выбор архитектуры сети для решения конкретной задачи основывается на опыте разработчика. Поэтому предложенная ниже архитектура сети является одним вариантом из множества возможных конфигураций. Для решения поставленной задачи сформируем трехслойную сеть обратного распространения, включающую 21 нейрон во входном слое (по числу компонент входного вектора) с передаточной функцией `logsig`, 15 нейронов во втором слое с передаточной функцией `logsig` и 3 нейрона в выходном слое (по числу компонентов выходного вектора) с передаточной функцией `purelin`. При этом в качестве обучающего алгоритма выбран алгоритм Levenberg-Marquardt (`trainlm`). Этот алгоритм обеспечивает быстрое обучение, но требует много ресурсов. В случае, если для реализации этого алгоритма не хватит оперативной памяти, можно использовать другие алгоритмы (`trainbfg`, `trainrp`, `trainscg`, `traincgb`, `traincgf`, `traincgp`, `trainoss`, `traingdx`). По умолчанию используется `trainlm`. Указанная сеть формируется с помощью процедуры:

```
net=newff(minmax(P),[21,15,3],{'logsig' 'logsig' 'purelin'},'trainlm');
```

Первый аргумент - матрица  $M \times 2$  минимальных и максимальных значений компонент входных векторов вычисляется с помощью процедуры `minmax`. Результатом выполнения процедуры `newff` является объект – нейронная сеть `net` заданной конфигурации. Сеть можно сохранить на диске в виде `mat`. файла с помощью команды `save` и загрузить снова с помощью команды `load`. Более подробную информацию о процедуре можно получить, воспользовавшись командой `help`.

## 3. Обучение сети



Следующий шаг – обучение созданной сети. Перед обучением необходимо задать параметры обучения. Задаем функцию оценки функционирования sse.

```
net.performFcn='sse';
```

В этом случае в качестве оценки вычисляется сумма квадратичных отклонений выходов сети от эталонов. Задаем критерий окончания обучения – значение отклонения, при котором обучение будет считаться законченным:

```
net.trainParam.goal=0.01;
```

Задаем максимальное количество циклов обучения. После того, как будет выполнено это количество циклов, обучение будет завершено:

```
net.trainParam.epochs=1000;
```

Теперь можно начинать обучение:

```
[net,tr]=train(net,P,T);
```

Таким образом, обучение сети окончено. Теперь эту сеть можно сохранить в файле nn1.mat:

```
save nn1 net;
```

#### 4. Тестирование сети

Перед тем, как воспользоваться нейронной сетью, необходимо исследовать степень достоверности результатов вычислений сети на тестовом массиве входных векторов. В качестве тестового массива необходимо использовать массив, компоненты которого отличаются от компонентов массива, использованного для обучения. В нашем случае для получения тестового массива достаточно воспользоваться еще раз программой mas1.

Для оценки достоверности результатов работы сети можно воспользоваться результатами регрессионного анализа, полученными при сравнении эталонных значений со значениями, полученными на выходе сети когда на вход поданы входные векторы тестового массива. В среде MATLAB для этого можно воспользоваться функцией postreg. Следующий набор команд иллюстрирует описанную процедуру:

```
>> mas1 - создание тестового массива P
```

```
>> y=sim(net,P); - обработка тестового массива
```

>> [m,b,r]=postreg(y(1,:),T(1,:)); - регрессионный анализ результатов обработки.

Сравнение компонентов С эталонных векторов с соответствующими компонентами выходных векторов сети. Если все точки легли на прямую, то это говорит о правильной работе сети на тестовом массиве.

>> [m,b,r]=postreg(y(2,:),T(2,:));

>> [m,b,r]=postreg(y(3,:),T(3,:));

Сохраним обученную сеть net на диске в файл nn1.mat save nn1 net

5. Моделирование сети. (Использование сети для решения поставленной задачи)

Для того, чтобы применить обученную сеть для обработки данных, необходимо воспользоваться функцией sim:

p=p';

Y=sim(net,p);

где p – набор входных векторов, Y – результат анализа в виде набора выходных векторов.

Например, пусть C=0.2, A=0.8, S=0.7,

Тогда

p=0.2\*exp(-((x-0.8).\*2/0.7));

Подставив этот входной вектор в качестве аргумента функции sim :

Y=sim(net,p)

Получим:

Y=0.2048 (C), 0.8150 (A), 0.7048 (S) >>

Что весьма близко к правильному результату (0.2; 0.8; 0.7).

*Лабораторная работа № 2*

### **Тема: Сети кластеризации и классификации данных**

В процессе анализа больших информационных массивов данных неизменно возникают задачи, связанные с исследованием топологической структуры данных, их объединением в группы (кластеры), распределением по классам и т. п. Это могут быть экономические, финансовые, научно-технические, медицинские и другие

приложения, где требуется решение таких практических задач, как сжатие данных, их хранение и поиск, определение характеристик объекта по ограниченному набору признаков. Такие задачи могут быть успешно решены с применением специального класса самоорганизующихся нейронных сетей.

### Самоорганизующиеся нейронные сети

Свойство самоорганизации является одним из наиболее привлекательных свойств нейронных сетей. Таким свойством обладают самоорганизующиеся нейронные сети, описанные финским ученым Т. Кохоненом. Нейроны самоорганизующейся сети могут быть обучены выявлению групп (кластеров) векторов входа, обладающих некоторыми общими свойствами. При изучении самоорганизующихся нейронных сетей, или *сетей Моина*, существенно различать сети с неупорядоченными нейронами, которые часто называют *слоями Кохонена*, и сети с упорядочением нейронов, которые часто называют *картами Кохонена*. Последние отражают структуру данных таким образом, что близким кластерам данных на карте соответствуют близко расположенные нейроны.

Для создания самоорганизующихся нейронных сетей, являющихся слоем или картой Кохонена, предназначены М-функции `newc` и `newsom` соответственно.

По команде `help selforg` можно получить следующую информацию об М-функциях, входящих в состав ППП Neural Network Toolbox и относящихся к построению сетей Кохонена:

<i>Self-organizing networks</i>	<i>Самоорганизующиеся сети</i>
<b>New networks</b>	<b>Формирование сети</b>
<code>newc</code> <code>newsom</code>	Создание слоя Кохонена
<b>Using networks</b>	<b>Работа с сетью</b>
<code>sim</code> <code>init</code> <code>adapt</code> <code>train</code>	Моделирование Инициализация
<b>Weight functions</b>	<b>Функции расстояния и взвешивания</b>
<code>negdist</code>	Отрицательное евклидово расстояние
<b>Net input functions</b>	<b>Функции накопления</b>

netsum	Сумма взвешенных входов
<b>Transfer functions</b>	<b>Функции активации</b>
compet	Конкурирующая функция активации
<b>Topology functions</b>	<b>Функции описания топологии сети</b>
gridtop hextop randtop	Прямоугольная сетка Гексагональная сетка
<b>Distance functions</b>	<b>Функции расстояния</b>
dist boxdist mandist linkdist	Евклидово расстояние Расстояние максимального координатного смещения Расстояние суммарного координатного смещения Расстояние связи
<b>Initialization functions</b>	<b>Функции инициализации сети</b>
initlay initwb initcon midpoint	Послойная инициализация Инициализация весов и смещений Инициализация смещений с учетом чувствительности нейронов Инициализация весов по правилу средней точки
<b>Learning functions</b>	<b>Функции настройки параметров</b>
learnk learncon learnsom	Правило настройки весов для слоя Кохонена Правило настройки смещений для слоя Кохонена Правило настройки весов карты Кохонена
<b>Adapt functions</b>	<b>Функции адаптации</b>
adaptwb	Адаптация весов и смещений
<b>Training functions</b>	<b>Функции обучения</b>
trainwb1	Повекторное обучение весов и смещений
<b>Demonstrations</b>	<b>Демонстрационные примеры</b>
democ1 demosm1 demosm2	Настройка слоя Кохонена Одномерная карта Кохонена Двумерная карта Кохонена

### *Слой Кохонена*

Рассмотрим самоорганизующуюся нейронную сеть с единственным слоем, задача которой заключается в том, чтобы правильно сгруппировать (кластеризовать) поступающие на нее векторы входа.

#### Создание сети

Для формирования слоя Кохонена предназначена М-функция `newsc`. Покажем, как она работает, на простом примере. Предположим, что задан массив из четырех двухэлементных векторов, которые надо разделить на 2 класса:

$$p = [.1 \ .8 \ .1 \ .9; \ .2 \ .9 \ .1 \ .8]$$

```
P =
    0.1000  0.8000  0.1000  0.9000
    0.2000  0.9000  0.1000  0.8000
```

В этом примере нетрудно видеть, что 2 вектора расположены вблизи точки (0,0) и 2 вектора - вблизи точки (1,1). Сформируем слой Кохонена с двумя нейронами для анализа двухэлементных векторов входа с диапазоном значений от 0 до 1:

```
net = newc([0 1; 0 1],2);
```

Первый аргумент указывает диапазон входных значений, второй определяет количество нейронов в слое. Начальные значения элементов матрицы весов задаются как среднее максимального и минимального значений, т. е. в центре интервала входных значений; это реализуется по умолчанию с помощью М-функции `midpoint` при создании сети. Убедимся, что это действительно так:

```
wts = net.IW {1,1}
```

```
wts =
    0.5000  0.5000
    0.5000  0.5000
```

Определим характеристики слоя Кохонена:

```
net.layers{1}
```

```
ans =
    dimensions: 2
    distanceFcn: 'dist'
    distances: [2x2 double]
    initFcn: 'initwb'
    netInputFcn: 'netsum'
    positions: [0 1]
    size: 2
    topologyFcn: 'hextop'
    transferFcn: 'compet'
```

userdata: [1x1 struct]

Из этого описания следует, что сеть использует функцию евклидова расстояния `dist`, функцию инициализации `initwb`, функцию обработки входов `netsum`, функцию активации `compnet` и функцию описания топологии `hextop`.

Характеристики смещений следующие:

```
net.biases{1}
ans =
    initFcn: 'initcon'
    learn:   1
    learnFcn: 'learncon'
    learnParam: [1x1 struct]
    size:     2
    userdata: [1x1 struct]
```

Смещения задаются функцией `initcon` и для инициализированной сети равны

```
net.b{1}
```

```
ans =
    5.4366
    5.4366
```

Функцией настройки смещений является функция `learncon`, обеспечивающая настройку с учетом параметра активности нейронов.

Элементы структурной схемы слоя Кохонена могут быть получены с помощью оператора `gensim(net)`

Они наглядно поясняют архитектуру и функции, используемые при построении слоя Кохонена.

Теперь, когда сформирована самоорганизующаяся нейронная сеть, требуется обучить ее решению задачи кластеризации данных. Напомним, что каждый нейрон блока `compnet` конкурирует за право ответить на вектор входа  $p$ . Если все смещения равны 0, то нейрон с вектором веса, самым близким к вектору входа  $p$ , выигрывает конкуренцию и возвращает на выходе значение 1; все другие нейроны возвращают значение 0.

*Правило обучения слоя Кохонена*

Правило обучения слоя Кохонена, называемое также *правилом Кохонена*, заключается в том, чтобы настроить нужным образом элементы матрицы весов.

Правило Кохонена представляет собой рекуррентное соотношение, которое обеспечивает коррекцию строки  $i$  матрицы весов добавлением взвешенной разности вектора входа и значения строки на предыдущем шаге. Таким образом, вектор веса, наиболее близкий к вектору входа, модифицируется так, чтобы расстояние между ними стало еще меньше. Результат такого обучения будет заключаться в том, что победивший нейрон, вероятно, выиграет конкуренцию и в том случае, когда будет представлен новый входной вектор, близкий к предыдущему, и его победа менее вероятна, тогда будет представлен вектор, существенно отличающийся от предыдущего. Когда на вход сети поступает все большее и большее число векторов, нейрон, являющийся ближайшим, снова корректирует свой весовой вектор. В конечном счете, если в слое имеется достаточное количество нейронов, то каждая группа близких векторов окажется связанной с одним из нейронов слоя. В этом и заключается свойство самоорганизации слоя Кохонена.

Настройка параметров сети по правилу Кохонена реализована в виде М-функции `learnk`.

#### *Правило настройки смещений*

Одно из ограничений всякого конкурирующего слоя состоит в том, что некоторые нейроны оказываются незадействованными. Это проявляется в том, что нейроны, имеющие начальные весовые векторы, значительно удаленные от векторов входа, никогда не выигрывают конкуренции, независимо от того как долго продолжается обучение. В результате оказывается, что такие векторы не используются при обучении и соответствующие нейроны никогда не оказываются победителями. Такие нейроны-неудачники называют *"мертвыми" нейронами*, поскольку они не выполняют никакой полезной функции. Чтобы исключить такую ситуацию и сделать нейроны чувствительными к поступающим на вход векторам, используются смещения, которые позволяют нейрону стать конкурентным с нейронами-победителями. Этому способствует положительное смещение, которое добавляется к отрицательному расстоянию удаленного нейрона.

Соответствующее правило настройки, учитывающее нечувствительность мертвых нейронов, реализовано в виде М-функции `learncon`.

Параметр скорости настройки по умолчанию равен 0.001, и его величина обычно на порядок меньше соответствующего значения для М-функции `learnk`. Увеличение смещений для неактивных нейронов позволяет расширить диапазон покрытия входных значений, и неактивный нейрон начинает формировать кластер. В конечном счете он может начать притягивать новые входные векторы.

Это дает два преимущества. Если нейрон не выигрывает конкуренции, потому что его вектор весов существенно отличается от векторов, поступающих на вход сети, то его смещение по мере обучения становится достаточно большим и он становится конкурентоспособным. Когда это происходит, его вектор весов начинает приближаться к некоторой группе векторов входа. Как только нейрон начинает побеждать, его смещение начинает уменьшаться. Таким образом, задача активизации "мертвых" нейронов оказывается решенной. Второе преимущество, связанное с настройкой смещений, состоит в том, что они позволяют выравнивать значения параметра активности и обеспечить притяжение приблизительно одинакового количества векторов входа. Таким образом, если один из кластеров притягивает большее число векторов входа, чем другой, то более заполненная область притянет дополнительное количество нейронов и будет поделена на меньшие по размерам кластеры.

#### *Обучение сети*

Реализуем 10 циклов обучения. Для этого можно использовать функции `train` или `adapt`:

```
net.trainParam.epochs = 10;
net = train(net,p) ;
net.adaptParam.passes = 10;
[net,y,e] = adapt(net,mat2cell(p));
```

Заметим, что для сетей с конкурирующим слоем по умолчанию используется обучающая функция `trainwb1`, которая на каждом цикле обучения случайно выбирает входной вектор предъявляет его сети; после этого производится коррекция весов и смещений.

Выполним моделирование сети после обучения:



```

a = sim(net,p);
ac = vec2ind(a)
ac = 2 1 2 1

```

Видим, что сеть обучена классификации векторов входа на 2 кластера: первый расположен в окрестности вектора (0, 0), второй - в окрестности вектора (1, 1). Результирующие веса и смещения равны:

```

wts1 = net.IW {1,1}
b1 = net.b {1}
wts1 =
    0.58383    0.58307
    0.41712    0.42789
b1 = 5.4152
    5.4581

```

Заметим, что первая строка весовой матрицы действительно близка к вектору (1, 1), в то время как вторая строка близка к началу координат. Таким образом, сформированная сеть обучена классификации входов. В процессе обучения каждый нейрон в слое, весовой вектор которого близок к группе векторов входа, становится определяющим для этой группы векторов. В конечном счете, если имеется достаточное число нейронов, каждая группа векторов входа будет иметь нейрон, который выводит 1, когда представлен вектор этой группы, и 0 в противном случае, или, иными словами, формируется кластер. Таким образом, слой Кохонена действительно решает задачу кластеризации векторов входа.

*Пример:*

Функционирование слоя Кохонена можно пояснить более наглядно, используя графику системы MATLAB. Рассмотрим 48 случайных векторов на плоскости, формирующих 8 кластеров, группирующихся около своих центров. На графике, приведенном на рис. 7.3, казано 48 двухэлементных векторов входа.

Сформируем координаты случайных точек и построим план их расположения шпоскости:

```

c = 8 ; n = 6 ; % Число кластеров, векторов в кластере
d = 0.5 ; % Среднеквадратичное отклонение от центра кластера
x = [-10 10;-5 5]; % Диапазон входных значений

```

```

[r,q] = size(x); minv = min(x')'; maxv = max(x')';
v = rand(r, c). * ((maxv - minv)*ones(1,c) + minv*ones(1,c));
t = c*n;% число точек
v=[v v v v v v]; v = v + randn(r,t)*d; % Координаты точек
P=v;
Plot(P(1,:), P(2,:),'+k')
Title('Векторы выхода', xlabel('P(1,:)'), ylabel('P(2,:)'))

```

После обучения в течение 500 циклов получим:

```

net.trainParam.epochs=500;
net=train(net,P);
w=net.IW{1}
bn=net.b{1}
cn=exp(1)./bn

```

Центры кластеризации распределились по восьми областям, смещения отклонились в обе стороны от исходного значения так же, как и параметры активности нейронов.

Рассмотренная самонастраивающаяся сеть Кохонена является типичным примером сети, которая реализует процедуру обучения без учителя.

### *Лабораторная работа № 3:*

#### **Тема: Сеть Хопфилда**

Всякий целевой вектор можно рассматривать как набор характерных признаков некоторого объекта. Если создать рекуррентную сеть, положение равновесия которой совпадало бы с этим целевым вектором, то такую сеть можно было бы рассматривать как ассоциативную память. Поступление на вход такой сети некоторого набора признаков в виде начальных условий приводило бы ее в то или иное положение равновесия, что позволяло бы ассоциировать вход с некоторым объектом. Именно такими ассоциативными возможностями и обладают *сети Хопфилда*. Они относятся к классу рекуррентных нейронных обладающих тем свойством, что за конечное число тактов времени они из произвольного начального состояния приходят в состояние устойчивого равновесия, называемое *аттракто-*

ром. Количество таких аттракторов определяет объем ассоциативной памяти сети Хопфилда.

Спроектировать сеть Хопфилда - это значит создать рекуррентную сеть со множеством точек равновесия, таких, что при задании начальных условий сеть в конечном счете приходит в состояние покоя в одной из этих точек. Свойство рекурсии проявляется в том, что выход сети подается обратно на вход.

Метод синтеза сети Хопфилда основан на построении системы линейных дифференциальных уравнений первого порядка, которая задана в некотором замкнутом гиперкубе пространства состояний и имеет решения в вершинах этого гиперкуба.

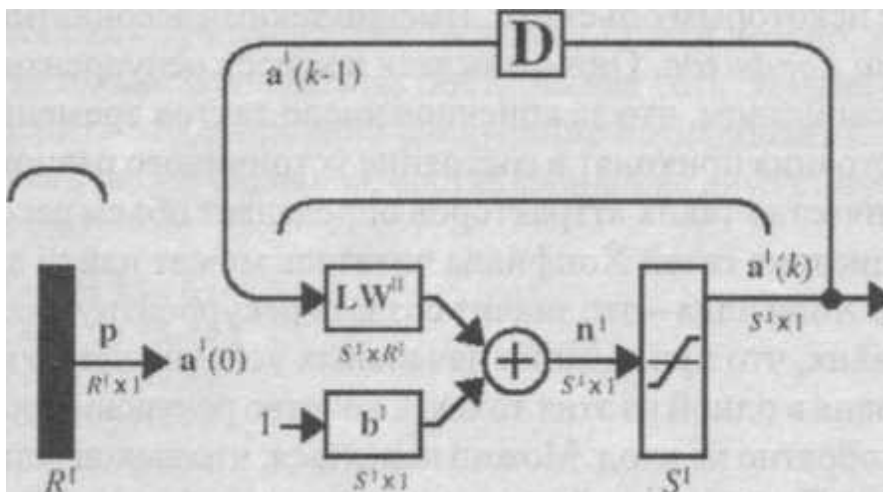
По команде `help hopfield` можно получить следующую информацию об М-функциях, входящих в состав ППП Neural Network Toolbox и относящихся к построению модифицированных сетей Хопфилда:

Hopfield recurrent networks	Рекуррентная модифицированная сеть Хопфилда
<b>New networks</b>	<b>Формирование сети</b>
nethop	Создание модифицированной сети Хопфилда
<b>Weight functions</b>	<b>Операции с весовой функцией</b>
dotprod	Скалярное произведение
<b>Net input functions</b>	<b>Операции над входами</b>
netsum	Суммирование
<b>Transfer functions</b>	<b>Функции активации</b>
satlins	Симметричная линейная функция с ограничениями
<b>Demonstrations</b>	<b>Демонстрационные примеры</b>
demohop1	Пример двумерной модифицированной сети Хопфилда Пример неустойчивой точки равновесия
demohop2 demo-hop3	Пример трехмерной модифицированной сети Хопфилда Пример устойчивых паразитных точек равновесия
demohop4	

*Архитектура сети*

Архитектура модифицированной сети Хопфилда представлена на рис.

1.



Начальные условия

Линейный слой с насыщением

Рис. 1.

Вход  $p$  устанавливает значения начальных условий. В сети используется линейная функция активации с насыщением  $satlins$ , которая описывается следующим образом:

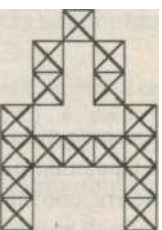
$$a = satlins(n) = \begin{cases} -1, & n < -1; \\ n, & -1 \leq n \leq 1; \\ 1, & n > 1. \end{cases}$$

Эта сеть может быть промоделирована с одним или большим количеством векторов входа, которые задаются как начальные условия. После того как начальные условия заданы, сеть генерирует выход, который по обратной связи подается на вход. Этот процесс повторяется много раз, пока выход не установится в положение равновесия. Можно надеяться, что каждый вектор выхода в конечном счете сойдется к одной из точек равновесия, наиболее близкой к входному сигналу.

### ***Распознавание образов***

#### *Постановка задачи*

Требуется создать нейронную сеть для распознавания 26 символов латинского алфавита. В качестве датчика предполагается использовать систему распознавания, которая выполняет оцифровку каждого символа, находящегося в поле зрения. В результате каждый символ будет представлен шаблоном размера  $5 \times 7$ . Например, символ А может быть представлен, как это показано на рис. 2.



0	0	1	0	0
0	1	0	1	0
0	1	0	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

Рис. 2.

Однако система считывания символов обычно работает неидеально и отдельные элементы символов могут оказаться искаженными (рис. 3).

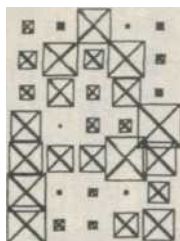


Рис. 3

Проектируемая нейронная сеть должна точно распознавать идеальные векторы входа и с максимальной точностью воспроизводить зашумленные векторы. М-функция `prprob` определяет 26 векторов входа, каждый из которых содержит 35 элементов, этот массив называется *алфавитом*. М-функция формирует выходные переменные `alphabet` и `targets`, которые определяют массивы алфавита и целевых векторов. Массив `targets` определяется как `eye(26)`. Для того чтобы восстановить шаблон для  $i$ -й буквы алфавита, надо выполнить следующие операторы:

```
[alphabet, targets] = prprob;
ti = alphabet(:, i);
letter{i} = reshape(ti, 5, 7)';
letter{i}
```

*Пример:*

Определим шаблон для символа А, который является первым элементом алфавита:

```
[alphabet, targets] = prprob;
i = 2;
ti = alphabet(:, i);
letter{i} = reshape(ti, 5, 7)';
letter{i}
ans =
    0 0 1 0 0
    0 1 0 1 0
    0 1 0 1 0
```

1 0 0 0 1  
1 1 1 1 1  
1 0 0 0 1  
1 0 0 0 1

### *Нейронная сеть*

На вход сети поступает вектор входа с 35 элементами; вектор выхода содержит 26 элементов, только один из которых равен 1, а остальные - 0. Правильно функционирующая сеть должна ответить вектором со значением 1 для элемента, соответствующего номеру символа в алфавите. Кроме того, сеть должна быть способной распознавать символы в условиях действия шума. Предполагается, что шум - это случайная величина со средним значением 0 и стандартным отклонением, меньшим или равным 0.2.

### *Архитектура сети*

Для работы нейронной сети требуется 35 входов и 26 нейронов в выходном слое. Для решения задачи выберем двухслойную нейронную сеть с логарифмическими сигмоидальными функциями активации в каждом слое. Такая функция активации выбрана потому, что диапазон выходных сигналов для этой функции определен от 0 до 1, и этого достаточно, чтобы сформировать значения выходного вектора.

Скрытый слой имеет 10 нейронов. Такое число нейронов выбрано на основе опыта и разумных предположений. Если при обучении сети возникнут затруднения, то можно увеличить количество нейронов этого уровня. Сеть обучается так, чтобы сформировать единицу в единственном элементе вектора выхода, позиция которого соответствует номеру символа, и заполнить остальную часть вектора нулями. Однако наличие шумов может приводить к тому, что сеть не будет формировать вектора выхода, состоящего точно из единиц и нулей. Поэтому по завершении этапа обучения выходной сигнал обрабатывается М-функцией `compet`, которая присваивает значение 1 единственному элементу вектора выхода, а всем остальным - значение 0.

### *Инициализация сети*

Вызовем M-файл `prprob`, который формирует массив векторов входа `alphabet` размера 35x26 с шаблонами символов алфавита и массив целевых векторов `targets`:

```
[alphabet,targets] = prprob;
```

```
[R,Q] = size (alphabet) ;
```

```
[S2,Q] = size (targets) ;
```

Двухслойная нейронная сеть создается с помощью команды `newff`:

```
S1 = 10;
```

```
net = newff(minmax(alphabet),[S1 S2], {'logsig' 'logsig'}, 'traingdx');
```

```
net.LW{2,1}=net.LW{2,1}*0.01;
```

```
net.b{2}=net.b{2}*0.01;
```

#### *Обучение*

Чтобы создать нейронную сеть, которая может обрабатывать зашумленные векторы входа, следует выполнить обучение сети как на идеальных, так и на зашумленных векторах. Сначала сеть обучается на идеальных векторах, пока не будет обеспечена минимальная сумма квадратов погрешностей. Затем сеть обучается на 10 наборах идеальных и зашумленных векторов. Две копии свободного от шума алфавита используются для того, чтобы сохранить способность сети классифицировать идеальные векторы входа. К сожалению, после того, как описанная выше сеть обучилась классифицировать сильно зашумленные векторы, она потеряла способность правильно классифицировать некоторые векторы, свободные от шума. Следовательно, сеть снова надо обучить на идеальных векторах. Это гарантирует, что сеть будет работать правильно, когда на ее вход будет передан идеальный символ. Обучение выполняется с помощью функции `trainbrx`, которая реализует метод обратного распространения ошибки с возмущением и адаптацией параметра скорости настройки.

#### *Обучение в отсутствие шума*

Сеть первоначально обучается в отсутствие шума с максимальным числом циклов обучения 5000 либо до достижения допустимой средней квадратичной погрешки равной 0.1:



```

P = alphabet;
T = targets;
net.performFcn = 'sse' ;
net.trainParam.goal = 0.1;
net.trainParam.show = 20;
net.trainParam.epochs = 5000;
net.trainParam.me = 0.95;
[net,tr] = train(net,P,T);

```

#### *Обучение в присутствии шума*

Чтобы спроектировать нейронную сеть, не чувствительную к воздействию шума, обучим ее с применением двух идеальных и двух зашумленных копий векторов алфавита. Целевые векторы состоят из четырех копий векторов. Зашумленные векторы имеют шум со средним значением 0.1 и 0.2. Это обучает нейрон правильно распознавать зашумленные символы и в то же время хорошо распознавать идеальные векторы.

При обучении с шумом максимальное число циклов обучения сократим до 300, а допустимую погрешность увеличим до 0.6:

```

netn = net;
netn.trainParam.goal = 0.6;
netn.trainParam.epochs = 300;
T=[targets targets targets targets];
for pass = 1:10
P=[alphabet, alphabet, ...
(alphabet + randn(R,Q)*0.1), ...
(alphabet + randn(R,Q)*0.2)];
[netn,tr] = train(netn,P,T);
end

```

#### *Повторное обучение в отсутствие шума*

Поскольку нейронная сеть обучалась в присутствии шума, то имеет смысл повторить ее обучение без шума, чтобы гарантировать, что идеальные векторы входа классифицируются правильно.

```
netn.trainParam.goal = 0.1; % Предельная среднеквадратичная погрешность
netn.trainParam.epochs = 500; % Максимальное количество циклов обучения
net.trainParam.show = 5; % Частота вывода результатов на экран
[netn,tr]=train(netn,P,T);
```

### *Эффективность функционирования системы*

Эффективность нейронной сети будем оценивать следующим образом. Рассмотрим 2 структуры нейронной сети: сеть 1, обученную на идеальных последовательностях, и сеть 2, обученную на зашумленных последовательностях. Проверка функционирования производится на 100 векторах входа при различных уровнях шума.

Приведем фрагмент сценария `appcr1`, который выполняет эти операции:

```
noise_range = 0:.05:.5;
max_test = 100;
network1 = [ ];
network2 = [ ];
T = targets;
% Выполнить тест
for noiselevel = noise_range
    errors1 = 0;
    errors2 = 0;
    for i=1:max_test
        P = alphabet + randn(35,26)*noiselevel;
        % Тест для сети 1
        A = sim(net,P);
        AA = compet(A);
        errors1 = errors1 + sum(sum(abs(AA-T)))/2;
        % Тест для сети 2
        An = sim(netn,P);
```

```

    AAn = compet(An);
    errors2 = errors2 + sum(sum(abs(AAn-T)))/2;
    echo off
    end
% Средние значения ошибок (100 последовательностей из 26 векторов
целей
network1 = [network1 errors1/26/100];
network2 = [network2 errors2/26/100];
end

```

Тестирование реализуется следующим образом. Шум со средним значением 0 и стандартным отклонением от 0 до 0.5 с шагом 0.05 добавляется к векторам входа. Для каждого уровня шума формируется 100 зашумленных последовательностей для каждого символа вычисляется выход сети. Выходной сигнал обрабатывается М-функцией `compet` с той целью, чтобы выбрать только один из 26 элементов вектора выхода. После этого оценивается количество ошибочных классификаций и вычисляется процент ошибки.

Сеть 1 обучена на идеальных векторах входа, а сеть 2 - на зашумленных. Обучение сети на зашумленных векторах входа значительно снижает погрешность распознавания реальных векторов входа. Сети имеют очень малые погрешности, если среднеквадратичное значение шума находится в пределах от 0.00 до 0.05. Когда к векторам был добавлен шум со среднеквадратичным значением 0.2, в обеих сетях начали возникать заметные ошибки. При этом погрешности нейронной сети, обученной на зашумленных векторах, на 3-4% ниже, чем для сети, обученной на идеальных входных последовательностях.

Если необходима более высокая точность распознавания, сеть может быть обучена либо в течение более длительного времени, либо с использованием большего количества нейронов в скрытом слое. Можно также увеличить размер векторов, чтобы пользоваться шаблоном с более мелкой сеткой, например 10x14 точек вместо 5x7.

## **8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ**

### **Практическая работа №1.**

#### **Обучение нейронной сети**

Качество работы нейронной сети сильно зависит от предъявляемого ей в процессе обучения набора учебных данных. Учебные данные должны быть типичными для задачи, решению которой обучается сеть. Обучение часто оказывается уникальным процессом, когда приемлемые решения многих проблем могут быть получены только в процессе многочисленных экспериментов. Разработчикам решения на основе нейронной сети требуется следующее.

- Выбрать соответствующую модель сети
- Определить топологию сети (т.е. число элементов и их связи)
- Указать параметры обучения

Часто разработчику необходимо выполнить и предварительную подготовку данных. Такая предварительная подготовка может быть совсем простой, - например, перевод с помощью масштабирования значений всех признаков (т.е. переменных) в диапазон от 0 до 1, - а может включать использование и бо-

лее сложных статистических процедур. Однако здесь следует подчеркнуть, что долгосрочной целью разработки нейронных сетей является минимизация необходимости прямого влияния разработчика на процесс нахождения решения, так как главным преимуществом нейронных сетей является их потенциальная возможность вырабатывать собственные решения. На практике лучшие результаты получаются тогда, когда имеется четкое понимание рассматриваемой проблемной области знаний и концептуальное понимание проблем построения нейронной сети. Данные, используемые для обучения нейронной сети, обычно разделяются на две категории: одни данные используются для обучения, а другие – для тестирования. На самом деле реальные качества нейронной сети являются только во время тестирования, поскольку успешное завершение обучения сети должно означать отсутствие признаков неправильной работы сети во время ее тестирования. Процесс тестирования разрабатывается так, чтобы в его ходе для данной сети можно было бы оценить ее способность обобщать полученные знания. Обобщение в данном случае означает способность сети правильно выполнять задачу с данными, которые оказываются хотя и аналогичными данным, предъявлявшимся сети в процессе обучения, но все же отличными от них.

### **Простой пример обучения**

Рассмотрим относительно простую задачу и выясним, как для ее решения можно использовать самый простой тип нейронной сети. Наша сеть будет состоять только из одного, входного и одного выходного элементов.

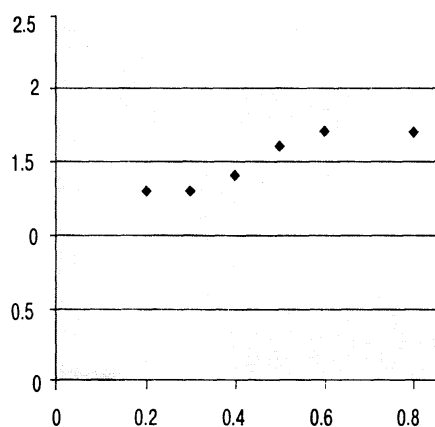
Многие в школе на уроке физики выполняли эксперимент, в котором требовалось измерить отклонение металлической пластины в зависимости от приложенной к ней нагрузки – гирь определенного веса. Потом необходимо было начертить график отклонения в зависимости от веса приложенной нагрузки. График строился по точкам, которые должны были в идеале образовать прямую линию. Используя метод наименьших квадратов, по полученным точкам можно построить прямую линию, с помощью которой можно будет предсказать, насколько отклонится пластина под нагрузкой, для которой отклонение не изме-

рялось. Многие проблемы моделируются с помощью построения прямых (или кривых) линий по имеющимся данным. Например, можно анализировать графики производства за прошедшие несколько лет, чтобы оценить, сколько стиральных машин изготовит фирма в течение следующих двух лет. Если тенденция изменения данных соответствует прямой линии, мы получим нечто похожее на диаграмму, показанную на рис. 1, где данные располагаются вдоль прямой, хотя и не существует прямой линии, на которой они лежали бы в точности. Этого и следовало ожидать – в реальном мире при любом измерении всегда имеются ошибки.

Уравнение прямой задается формулой

$$y = tx + c,$$

где  $y$  и  $x$  являются переменными (например, отклонением и нагрузкой),  $t$  определяет наклон или градиент прямой, а  $c$  - значение сдвига (т.е. точку, в которой прямая пересекает ось  $y$ ).



*Рис. 1. Данные располагаются вдоль прямой линии, но не лежат в точности на прямой из-за ошибок измерения*

Можно провести прямую на глаз, а затем измерить ее наклон и значение сдвига, но метод наименьших квадратов дает нам возможность вычислить тис. Что означает "наиболее подходящая прямая"? В данном случае это прямая, для которой сумма квадратов ошибок для всех точек, соответствующих имеющимся данным, оказывается наименьшей. Что такое ошибки для набора точек, показано на рис. 1. Чтобы найти сумму квадратов ошибок, следует возвести значение каждой из ошибок в квадрат и просуммировать все полученные таким образом значения.

При использовании метода наименьших квадратов  $t$  находится по формуле

$$m = \frac{n \sum x_i y_j - \sum x_i \sum y_j}{n \sum x_i^2 - (\sum x_i)^2} \quad (1)$$

а  $c$ - по формуле

$$c = \frac{\sum y_i - m \sum x_i}{n} \quad (2)$$

где  $x_i$  и  $y_j$  представляют значения координат для точки  $i$ , а  $n$  равно числу точек. Суммирование выполняется по всем точкам данных.

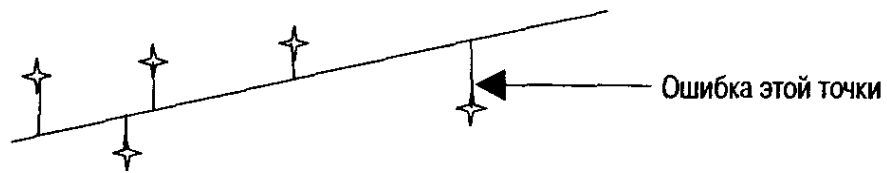


Рис. 2. Каждой точке соответствует своя ошибка, равная расстоянию от точки до прямой

### Вывод уравнений для $m$ и $c$

Как только с данными оказывается сопоставленной некоторая прямая, с помощью уравнения этой прямой для любого данного значения  $x$  можно получить оценку соответствующего значения  $y$ . В реальности любой оценке значения  $y$  соответствует своя ошибка. Поэтому для оценки  $y$  мы можем записать:

$$y_i = mx_i + c + e_i \quad (3)$$

где  $e_i$  обозначает ошибку  $i$ . Суммирование квадратов ошибок,  $E$ , вычисляется по формуле

$$E = \sum e_i^2 \quad (4)$$

Используя равенства (3) и (4), получаем:

$$E = \sum [y_i - (mx_i + c)]^2 \quad (5)$$

Теперь, рассмотрев частные производные уравнения (5), мы увидим, как зависит общая ошибка от изменения  $m$  и  $c$ :

$$\frac{\partial E}{\partial m} = -2 \sum x_i [y_i - (mx_i + c)], \quad (6)$$

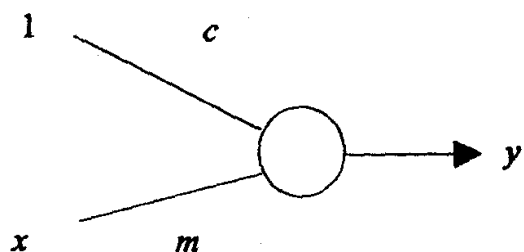
$$\frac{\partial E}{\partial c} = \sum x_i [y_i - (mx_i + c)], \quad (7)$$

Если (6) и (7) приравнять к 0 и решить соответствующие уравнения, будут получены равенства (1) и (2).

Метод наименьших квадратов дает эффективный способ нахождения прямой, наилучшим образом соответствующей имеющемуся набору данных.

Метод прост в применении, но его обоснование требует определенного уровня знания математики. Для нахождения прямой, соответствующей имеющемуся набору данных, можно использовать нейронную сеть. Такой сети нужно просто предоставить учебные данные и дать возможность обучиться на них.

Сеть с одним входным и одним выходным элементами была обучена проводить прямую линию на основе анализа имеющихся данных (см. рис. 1.). В этой сети использовалась линейная функция активности. Задача требовала, чтобы сеть оценивала тис, поэтому тис являются параметрами сети (т.е. весовыми коэффициентами), значения для которых перед началом обучения были выбраны случайным образом из диапазона между -0.3 и +0.3. Модель сети показана на рис. 3. Данными для обучения были значения координаты  $x$  для каждой точки и соответствующие целевые значения координаты  $y$ . Вес  $c$  на входе имеет значение 1 ( $c$ , умноженное на 1, равно  $c$ ; этот вес задает смещение). Для обучения сети использовалось дельта-правило, а норма обучения была выбрана равной 0.1. Обучение заканчивалось после рассмотрения каждой точки 10000 раз.



*Рис. 3. Линейный элемент, который можно обучить найти прямую для данных на рис. 4*

Оценки для  $m$  и  $c$ , полученные с помощью сети, в результате применения метода наименьших квадратов, представлены в следующей таблице, а линия, найденная сетью, показана на рис. 4.



Параметр	Метод наименьших квадратов	Сеть
$m$	1.0085	1.0284
$c$	1.0450	1.0360

На самом деле совсем не удивительно, что сеть по сравнению с методом наименьших квадратов дает сравнимые результаты, поскольку, дельта-правило обучения выводится из принципа минимизации суммы квадратов ошибок. Наша сеть решила данную задачу путем обобщения результатов вывода в зависимости от ввода, подобрав для них некоторую линейную зависимость. В нашем случае сеть моделирует прямую линию, но большие нейронные сети с нелинейными функциями активности могут подбирать для данных обучения весьма сложные формы и таким образом решать многие весьма сложные задачи.

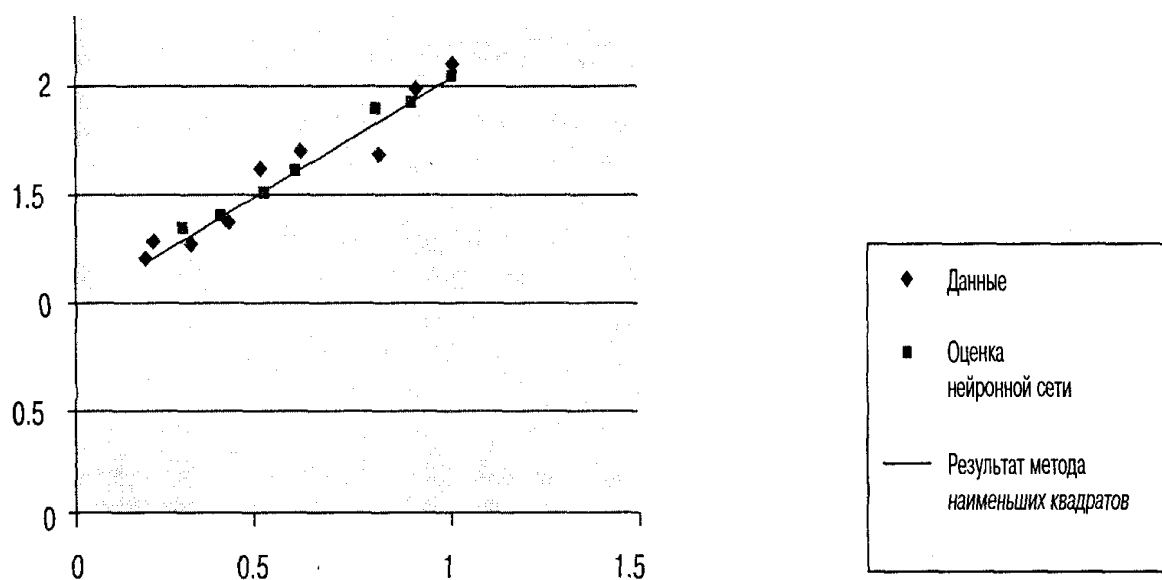


Рис. 4. Данные и соответствующая прямая, найденная с помощью нейронной сети, схема которой показана на рис. 3. Найденная нейронной сетью прямая мало отличается от прямой, получаемой при использовании метода наименьших квадратов

### Упражнения

1. Для данных, представленных ниже, начертите на глаз несколько прямых, которые могут соответствовать этим данным. Запишите уравнения этих прямых, измерив соответствующие наклоны и координаты точек пересече-

ния прямых с осью  $y$ . Для каждой прямой вычислите среднеквадратическую ошибку при условии, что для вводимых значений  $x$  вывод задается формулой  $вывод = tx + c$

$x$	Требуемый вывод
0,3	1,6
0,35	1,4
0,4	1,4
0,5	1,6
0,6	1,7
0,8	2
0,95	1,7
1,1	2,1

2. Для данных из упражнения 1 найдите прямую, получаемую в результате применения метода наименьших квадратов.

3. Для данных упражнения 1 и заданных начальных весовых коэффициентов  $вывод = 0.5x + 0.5$  вычислите новую прямую после одного прохода через данные, используя правило обучения Видроу-Хоффа (дельта-правило) с нормой обучения, равной 0.3. (Замечание: после рассмотрения каждого учебного образца получается новая прямая.)

4. Наша подбирающая прямые линии нейронная сеть имела один входной элемент, а целью было нахождение весовых коэффициентов, при которых по заданному  $x$  можно было бы оценить  $y$ . Во многих задачах требуется знать, с какой стороны от прямой (т.е. выше или ниже ее) будет располагаться точка данных. Представьте полученную в результате решения упражнения 2 прямую в следующем виде:

$$ввод(x, y) = tx - y + c$$

Вычислите ввод для всех точек данных, определенных в формулировке упражнения 1, и для каждой точки найдите выходное значение, используя двоичную пороговую функцию. Как вы думаете, можно ли моделировать с помощью нейронной сети вычисление значений ввода?

*Практическая работа № 2*

**Классификация образцов**

Функция выбора решения

Рассмотрим тривиальную задачу и используем для решения этой задачи самый простой тип сети.

Иллюстрация такой задачи классификации представлена на рис. 1. Задача состоит в выработке правил классификации самолетов для бомбардировщиков и истребителей в зависимости от их максимальной скорости и максимального взлетного веса. Такие правила могут быть заданы формально:

ЕСЛИ вес  $> 0.80$  И скорость  $< 0.55$ , ТО бомбардировщик,

ЕСЛИ вес  $< 0.90$  И скорость  $> 0.25$ , ТО истребитель.

Эти правила используют дискретные граничные значения, разделяющие пространство всех значений на прямоугольные области. Разделение, порожденное этими правилами, вполне успешно классифицирует самолеты, представленные на нашей диаграмме, но оказывается не слишком гибким, если по этим правилам придется классифицировать новый самолет. Кроме того, эти правила в указанном виде ничего не сообщают о том, насколько точной будет классификация нового самолета.

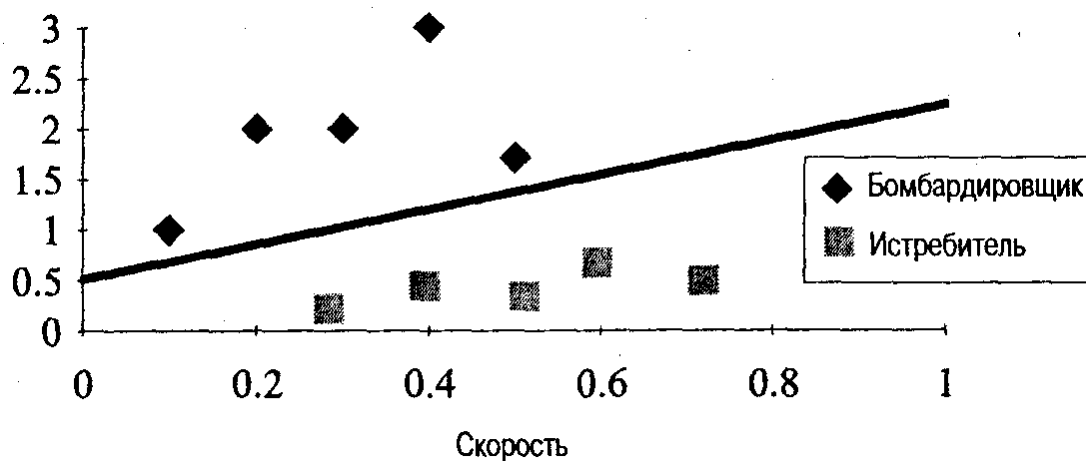


Рис. 1. Разделение абстрактных данных на два класса

Альтернативный подход к использованию правил заключается в выводе функции классификации путем построения прямой, разделяющей два класса. Для нового самолета нам нужно просто указать точку на плоскости, соответствующую известным значениям максимальной скорости и максимального взлетного веса и посмотреть, по какую сторону от прямой будет расположена эта точка. В данном случае для небольшого количества самолетов рассмотрены два признака,

скорость и вес, поэтому можно представить данные в виде изображения на плоскости. Однако, если придется иметь дело с сотнями самолетов и значительно большим числом признаков (т.е. в случае многомерной задачи), задачу классификации в виде простой картинке представить будет невозможно.

Выход заключается в использовании функции выбора решения. Уравнение прямой, разделяющей два типа самолетов, записывается в следующем виде:

$$x_2 = 1.5 x_1 + 0.5,$$

где  $x_1$  представляет скорость, а  $x_2$  — вес. Это уравнение можно использовать для создания функции выбора решения:

$$f(x_1, x_2) = -x_2 + 1.5x_1 + 0.5,$$

$$d = \begin{cases} \text{истребитель,} & \text{если } f(x_1, x_2) \geq 0, \\ \text{бомбардировщик,} & \text{если } f(x_1, x_2) \leq 0. \end{cases}$$

Например, истребитель, представленный точкой (0.4, 0.5), даст

$$f(0.4, 0.5) = -0.5 + 1.5 * 0.4 + 0.5 = 0.6,$$

и функция выбора решения правильно классифицирует эту точку, как истребитель.

Предлагаемую функцию выбора решения можно моделировать с помощью нейронной сети и даже реализовать в виде аппаратных средств. На рис. 2 показана сетевая модель для данной функции выбора решения. Сетевой ввод для центрального элемента находится путем умножения переменных ввода,  $x_1$  и  $x_2$ , на соответствующие их взвешенным связям коэффициенты с последующим суммированием результатов. Указанное на схеме значение смещения тоже добавляется в сумму, в результате чего получается значение комбинированного ввода для данного элемента. Вся сумма имеет следующий вид:

$$u_j = w_0 + \sum_{i=1}^n x_i w_{ij} \text{ где } u_j \text{ представляет значение комбинированного ввода, } w_0$$

— смещение, связываемое с элементом, значение активности которого считается всегда равным 1,  $x_i$  — значение активности  $i$ -го элемента, а  $w_{ij}$  — вес связи, ведущей от элемента с номером  $i$  к элементу с номером  $j$ . Для элемента,

показанного на рис. 2, выходное значение вычисляется согласно критерию пороговой функции:

$$\text{вывод} = \begin{cases} 1, & \text{если комбинированный ввод} \geq 0 \\ 0, & \text{если комбинированный ввод} < 0 \end{cases}$$

Выходное значение 1 должно указывать на то, что самолет является истребителем, а значение 0 должно соответствовать бомбардировщику.

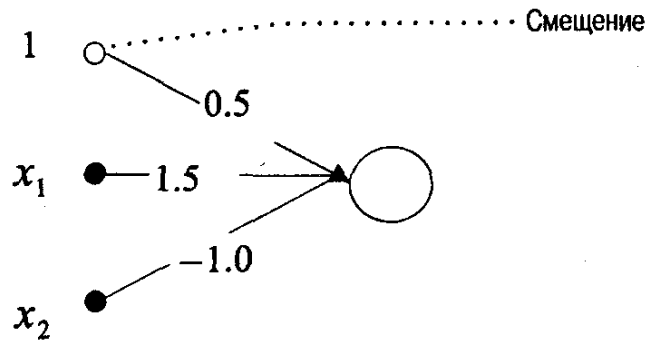


Рис. 2. Простая нейронная сеть

Ввод для сети на рис. 2 пропускается через пороговую функцию, в результате чего порождается выходное значение, равное 0 или 1. Вместо указанной пороговой функции можно было бы использовать любую другую функцию, дающую выходные значения в диапазоне от 0 до 1. Так, если выходное значение оказывается равным 0.9, мы можем быть практически уверены, что самолет является истребителем, а вот значение 0.5 не дает нам уверенности относительно его классификации. На самом деле систему можно использовать не только для классификации, а и для оценки степени точности такой классификации.

### Пример 1

(а) Вычислите комбинированный сетевой ввод для элемента на рис. 2 и соответствующее выходное значение при использовании пороговой функции и входного вектора [0.7 2.5].

(б) Вычислите выходное значение, использовав в качестве функции активности сигмоидальную функцию. Входной вектор остается таким же, как и в п. (а).

(в) Вычислите комбинированный ввод для сети с архитектурой, показанной на рис. 2, но с набором весовых значений  $[-0.2 \ 0.03 \ 1.2]$  и таким же входным вектором, как и в п. (а).

### Решение 1

(а) Порядок элементов во входном векторе говорит о том, что  $x_1=0.7$  и  $x_2=0.5$ . Таким образом, комбинированный ввод оказывается равным

$$0.5 + (0.7*1.5) + (2.5*-1) = -0.95 .$$

Комбинированный ввод оказывается отрицательным, поэтому вывод равен 0.

(б) Активность в случае сигмоидальной функции вычисляется по следующей формуле:

$$f(u_j) = \frac{1}{1 + \exp(-u_j)}$$

Комбинированный ввод равен  $-0.95$ , и подстановка этого значения в сигмоидальную функцию дает выходное значение  $0.28$ .

(в) Порядок размещения весовых значений в матрице весов означает, что смещение равно  $(-0.2)$ , а для элемента, от которого исходит смещение, значение активности должно быть равным 1 . Если входной вектор обозначить  $x$ , а вектор весов —  $w$ , то комбинированный ввод элемента можно выразить в виде

$$u_j = xw$$

при условии, что входной вектор включает и значение активности элемента смещения. Добавляя к входному вектору значение активности элемента смещения, для комбинированного ввода в нашем случае получаем

$$u = [1 \ 0.7 \ 2.5] \cdot \begin{bmatrix} -0.2 \\ 0.03 \\ 1.2 \end{bmatrix} = (1 * -0.2) + (0.7 * 0.03) + (2.5 * 1.2) = 2.82$$

### Пример 2

Найти весовые коэффициенты для модели нейронной сети, подобной показанной на рис. 2 и представляющей следующее уравнение:

$$2x_2 = -4x_1 + 8.$$

### Решение 2

Для любой точки, лежащей на указанной прямой, весовые коэффициенты можно определить из уравнения

$$w_0 + x_1 w_1 + x_2 w_2 = 0$$

Отсюда получаем

$$x_2 = -x_1 \frac{w_1}{w_2} - \frac{w_0}{w_2}$$

Сравнивая члены полученного равенства с коэффициентами, указанными в условии примера, имеем

$$-\frac{w_1}{w_2} = -\frac{4}{2} \quad -\frac{w_0}{w_2} = -\frac{8}{2}$$

Таким образом,  $w_0 = -8$ , а  $w_2 = -2$ .

#### Корректировка весов

Из рис. 1 должно быть ясно, что имеется целое множество прямых, которые могут быть выбраны в качестве границы, разделяющей данные, поэтому имеется целое множество весовых значений, дающих подходящее решение. Если требуемый выход  $i$ -го элемента обозначить  $t$ , а наблюдаемый на самом деле —  $o_j$ , то ошибка  $E_p$  для образца  $p$  может быть определена по формуле

$$E_p = \frac{1}{2} \sum_j (t_j - o_j)^2, \quad (1)$$

а полная ошибка будет равна  $E = \sum E_p$ . Множитель  $1/2$  включен здесь в формулу с целью упрощения выкладок.

Активность любого элемента зависит от комбинированного ввода этого элемента, а значит, от весовых значений, влияющих на этот элемент. Представьте себе элемент, подобный показанному на рис. 2, но без смещения. Такой элемент может моделировать любую прямую, проходящую через начало координат. В случае линейного элемента и одного образца равенство (1) можно записать в виде

$$E = \frac{1}{2} (t - u)^2,$$

так как для линейного элемента вывод оказывается равным вводу. Разворачивание правой части равенства дает

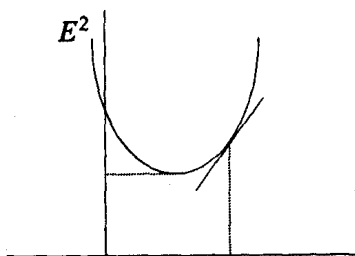
$$E = \frac{1}{2}[t^2 - 2tu + u^2] = \frac{1}{2}[t^2 - 2t(x_1w_1 + x_2w_2) + x_1^2w_1^2 + 2x_1w_1x_2w_2 + x_2^2w_2^2], \quad (2)$$

где  $u = x_1w_1 + x_2w_2$ . Дифференцируя равенство (2) по  $w_1$ , получаем

$$\frac{\partial E}{\partial w_1} = (-t + x_1w_1 + x_2w_2)x_1. \quad (3)$$



Равенство (2) говорит о том, что зависимость квадрата ошибки от  $w_1$  является параболической, как показано на рис. 3, а если приравнять к нулю правую часть равенства в (3) и решить полученное таким образом уравнение, можно найти точку минимума соответствующей кривой.



*Рис. 3. Прямая линия представляет производную ошибки в зависимости от веса в момент времени  $n$*

В процессе обучения сеть должна корректировать весовые коэффициенты так, чтобы максимально уменьшить значение общей ошибки. Другими словами, весовые коэффициенты должны корректироваться в том направлении, в котором спуск вниз по поверхности ошибок происходит быстрее всего. На рис. 3 эта идея иллюстрируется для одного веса, где  $n$  обозначает время или итерацию.

### **Минимизация квадрата ошибки**

Нанесем на координатную сетку точки, соответствующие величине изгиба металлической пластины в зависимости от приложенной к ней нагрузки, с последующим построением прямой, для которой сумма квадратов расстояний от полученных точек до прямой оказывалась минимальной. Этот же принцип можно использовать для корректировки весов, и одно из таких правил корректировки весов, называемое правилом Видроу—Хоффа или дельта-правилом. Это правило записывается в следующем виде:

$$\Delta w_{ij} = \eta \delta_j x_i, \quad \delta_j = (t_j - o_j), \quad (5)$$

где  $t_j$  обозначает требуемое значение для элемента  $j$ ,  $o_j$  — его реальный вывод,  $x_i$  — сигнал, приходящий от элемента  $i$ ,  $\eta$  — норма обучения (коэффициент, от которого зависит величина изменения веса), а  $\Delta w_{ij}$  — величина, на которую изменяется вес для связи, идущей от элемента  $i$  к элементу  $j$ .

Это правило очень просто получается в случае линейного элемента, когда вывод определяется следующей формулой:

$$o_j = \sum_i x_i w_{ij}$$

Используя цепное правило, можно выразить производную поверхности ошибок в зависимости от веса в виде произведения, характеризующего изменение ошибки в зависимости от вывода элемента, и изменение вывода в зависимости от связанных с элементом весовых коэффициентов:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial w_{ij}}, \quad (6)$$

$$\frac{\partial E}{\partial o_j} = -\delta_j \quad (\text{это следует из (1) и определения } \delta_j \text{ в (5)}),$$

$$\frac{\partial o_j}{\partial w_{ij}} = x_i$$

откуда, возвращаясь к (6), получаем

$$\frac{\partial E}{\partial w_{ij}} = \delta_j x_i$$

Принимая во внимание тот факт, что вес должен изменяться в направлении, противоположном направлению вектора градиента, и умножая на норму обучения, приходим к равенству (5).

### **Упражнения**

1. На рис. 4 показана сеть с обратным распространением ошибок во время обработки учебного вектора  $[1.0 \ 0.9 \ 0.9]$ , для которого целевым выходным вектором является  $[0.1 \ 0.9 \ 0.1]$ . Пусть выходом элемента В является значение 0.6, а выходом элемента С — значение 0.8. Предположим, что функцией активности является сигмоид.
  - (a) Вычислите фактический выходной вектор.
  - (b) Вычислите изменения весовых значений для связей, идущих от элемента А. Норма обучения предполагается равной 0.25.

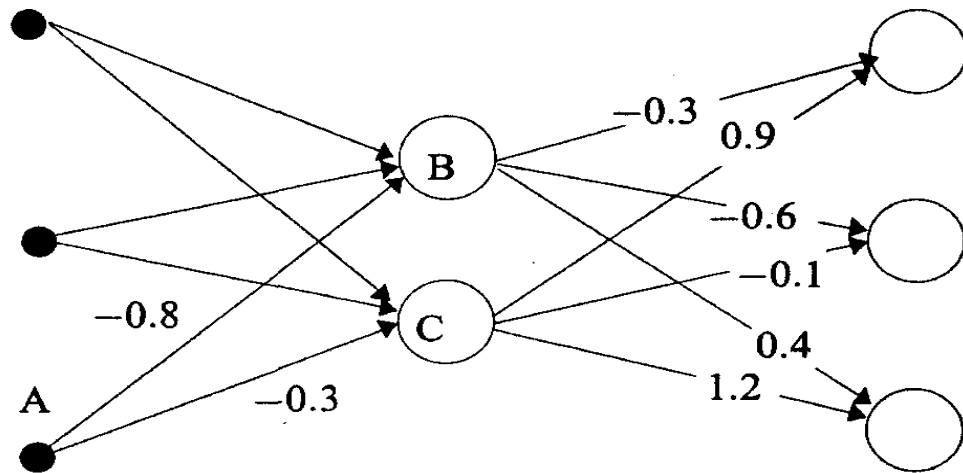


Рис. 4. Сеть типа 3-2-3, в которой нет элементов смещения

2. Повторите вычисления упражнения 1 для целевого выходного вектора [0.1 0.9 0.9].

### Практическая работа № 3

#### Тема: Алгоритм обратного распространения ошибок

На первой стадии происходит инициализация весов малыми случайными значениями — например, значениями из диапазона между -0.3 и +0.3. Обучение предполагается управляемым, поскольку с каждым входным образцом связывается целевой выходной образец. Обучение продолжается до тех пор, пока изменение усредненной квадратичной ошибки не окажется меньше некоторого допустимого значения при переходе от одной эпохи к следующей. Например, допустимое значение 0.01 означает, что усредненная квадратичная ошибка соседних эпох не должна отличаться более чем на  $\pm 0.01$ . Если в процессе обучения наступает момент, когда ошибка в сети попадает в рамки допустимого изменения, говорят, что наблюдается сходимость. Другим критерием окончания обучения можно считать наступление момента, когда выход для каждого учебного образца оказывается в рамках допустимого отклонения от соответствующего целевого выходного образца.

Чтобы уменьшить вероятность того, что изменения весов приобретут осциллирующий характер, вводится инерционный член  $\alpha$ , добавляемый в пропорции, соответствующей предыдущему изменению веса:

$$\Delta w_{ij}(n+1) = \eta \delta_j o_i + \alpha \Delta w_{ij}(n)$$

Таким образом, изменение веса на шаге  $n + 1$  оказывается зависящим от изменения веса на шаге  $n$ . Алгоритм обратного распространения в целом представлен на рис. 1.

### Пример 1

Вводимыми данными будут  $(0.1, 0.9)$ , норма обучения равна  $0.8$ , а коэффициент инерции равен  $0$ . Сеть имеет архитектуру  $2-2-2-1$ , а весовые значения задаются матрицами

$$\begin{bmatrix} 2 & 2 \\ -2 & 3 \\ -2 & 3 \end{bmatrix}, \begin{bmatrix} 3 & -2 \\ -2 & 2 \\ -4 & 2 \end{bmatrix}, \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix},$$

Представьте полностью прямой и обратный проходы в сети с прямой связью, использующей алгоритм обратного распространения ошибок.

### Решение примера 1:

На рис. 3 показан прямой поток данных в сети с указанными суммарным вводом и суммарным выводом каждого элемента. Обратный поток ошибок показывает взвешенные ошибки, получаемые элементами, и общую ошибку, вычисляемую по формуле, представленной на рис. 2.

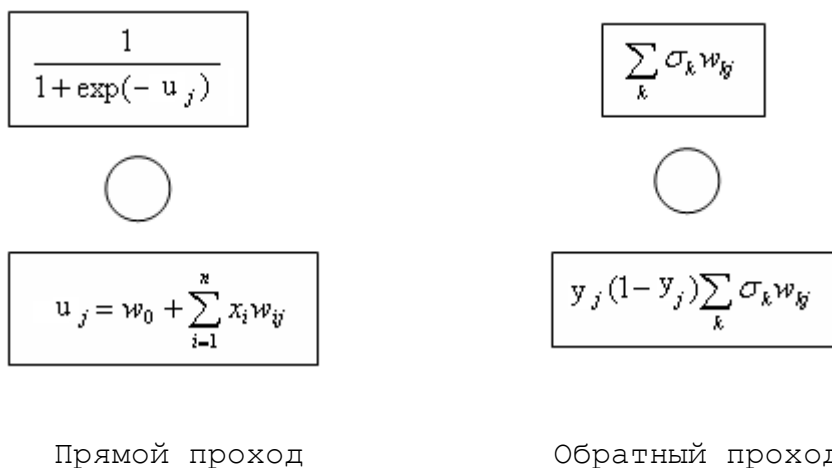


Рис. 2. Числа перед элементами и после них на рис. 3 соответствуют вычислениям по формулам, заключенным здесь в рамки

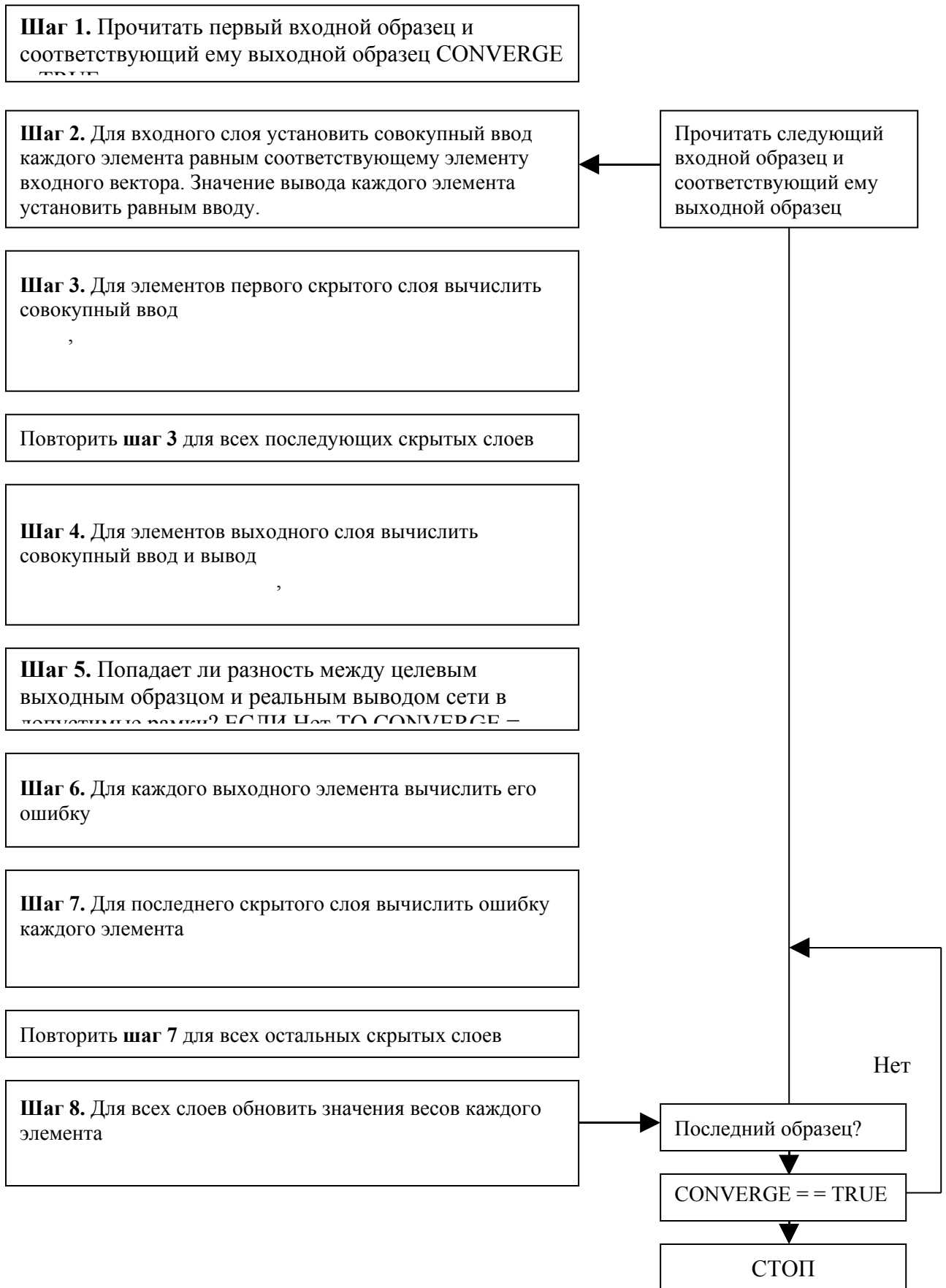


Рис. 1. Алгоритм обратного распространения ошибок. Индекс  $k$  соответствует предыдущему слою при обратном движении в сети.

## Пример 2.

Представьте полностью прямой и обратный проходы в сети с прямой связью, использующей алгоритм обратного распространения ошибок, для входного образца [0.1 0.9] и целевого выходного значения 0.9 в предположении, что сеть имеет архитектуру 2-2-1 (т.е. два входных, два скрытых один выходной элемент) с весовыми коэффициентами

$$\begin{bmatrix} 0,1 & 0,1 \\ -0,2 & -0,1 \\ 0,1 & 0,3 \end{bmatrix} \text{ для первого слоя, } \begin{bmatrix} 0,2 \\ 0,2 \\ 0,3 \end{bmatrix} \text{ для второго слоя.}$$

Выход для входных элементов совпадает с вводимыми значениями. Первые строки обеих весовых матриц определяют элементы смещения соответствующего слоя, которые, как вы помните, связываются с элементами, значения активности которых равны 1. Элементы получают номера {0,1,2} для входного слоя, {3,4,5} — для скрытого слоя и {6} — для выходного слоя. При этом элементы с номерами 0 и 3 оказываются элементами смещения для входного и скрытого слоев соответственно.

## Практическая работа № 4

### Тема: Кластеризация образцов

При обучении без управления сети приходится проводить кластеризацию образцов (т.е. разделение их на группы) самостоятельно. Все образцы одного кластера должны иметь что-то общее – они будут оцениваться, как подобные. Предположим, например, что перед нами стоит задача классификации мебели по признакам полезности и красоты. Все объекты, подобное стулу, попадут при этом в одну группу, а все объекты, подобные столу, – в другую. Эти группы затем анализируются, и от группы подобных столу предметов отделяется группа письменных столов. Группа письменных столов подобна группе предметов, подобных столам, поэтому эти группы должны разместиться близко одна к другой и далеко от группы предметов, подобных стулу.

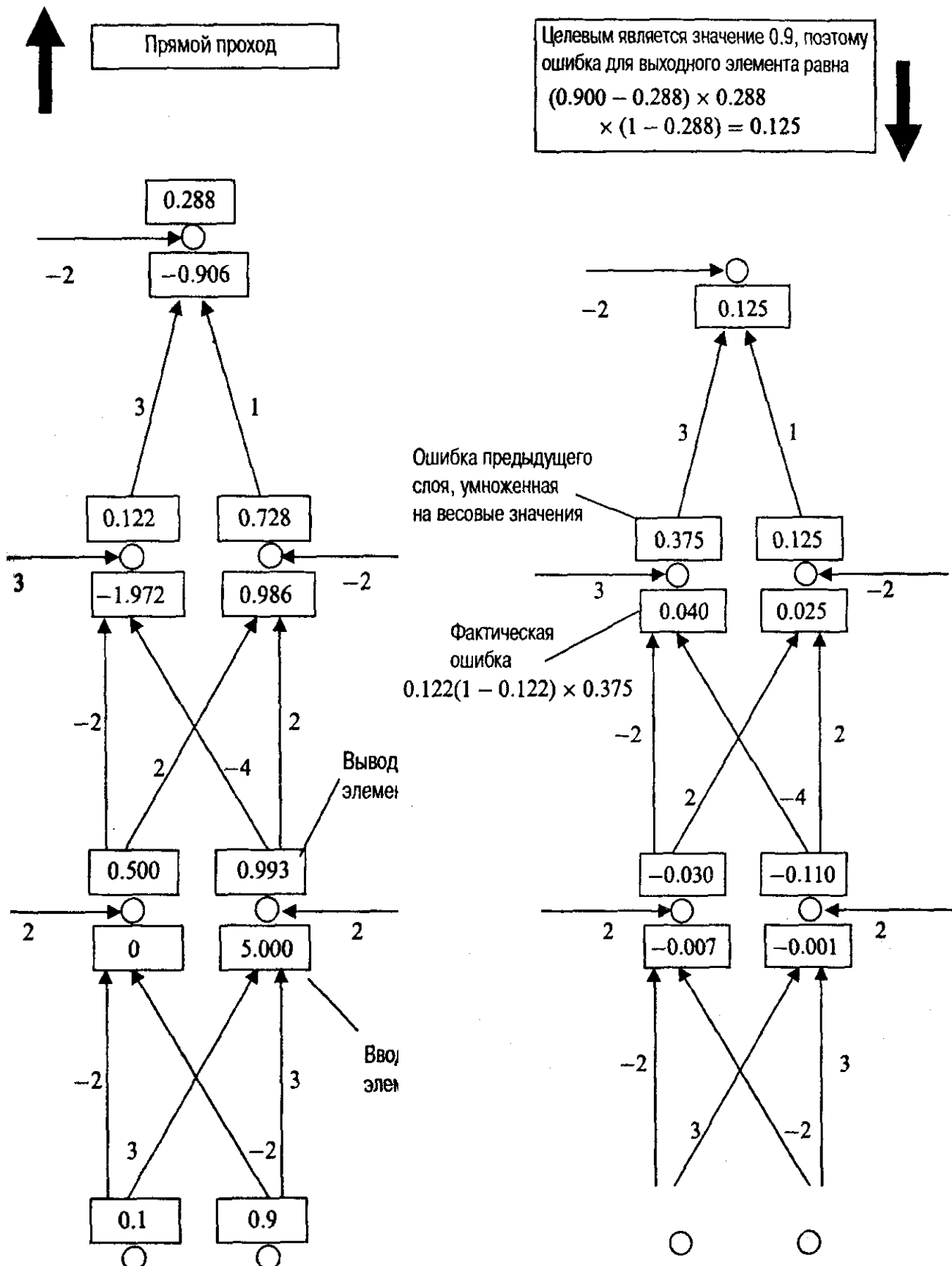
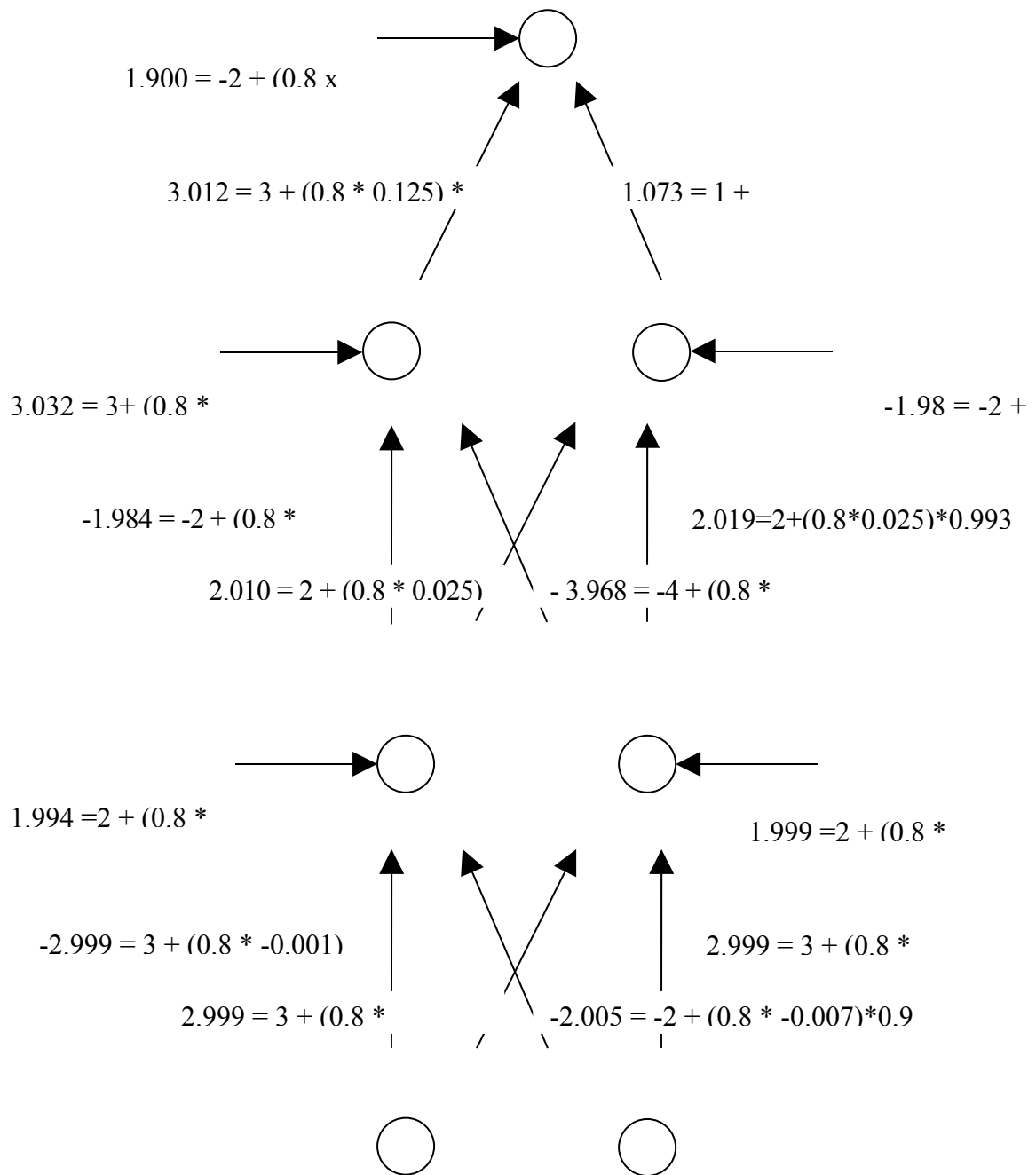


Рис. 3. Пример прямого и обратного проходов в сети типа 2-2-2-1 с прямой связью. Данные ввода, вывода и значения ошибок показаны в рамках (см. рис. 2)



*Рис. 4. Новые весовые значения, вычисленные по значениям ошибок, показанных на рис. 3*

Алгоритмы кластеризации выполняют такие операции с образцами данных. Группы в дальнейшем мы будем называть кластерами и предполагать, что разделение образцов на кластеры должно удовлетворять следующим двум требованиям

- Образцы внутри одного кластера должны быть в некотором смысле



подобны.

•Кластеры, подобные в некотором смысле, должны размещаться близко один от другого.

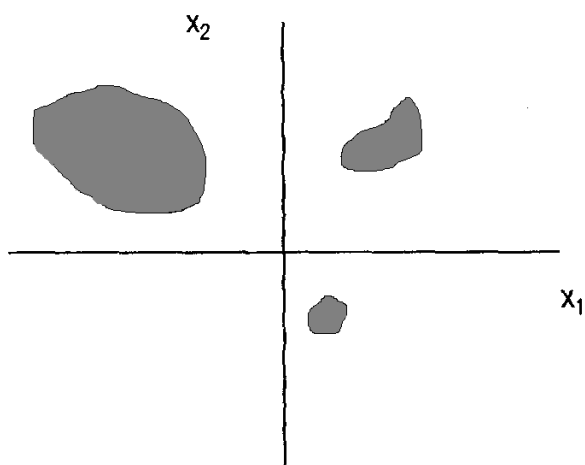


Рис. 1. Данные, формирующие три кластера

На рис. 1 показано размещение на двумерной плоскости данных, которые естественным образом организуются в три кластера: соответствующая образцу точка попадает в определенный кластер, если она располагается близко к точкам этого кластера в сравнении с точками, принадлежащими другим кластерам. Мерой близости (или подобия) двух точек обычно является квадрат евклидова расстояния между ними, вычисляемый по формуле

$$d_{pq} = \sum_i^n (x_{pi} - x_{qi})^2,$$

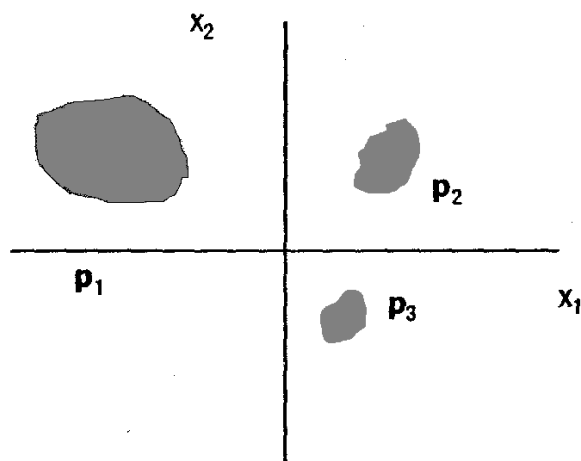
где  $d_{pq}$  обозначает квадрат евклидова расстояния между точкой  $p$  и точкой  $q$ ,  $x_{pi}$  —  $i$ -я координата образца  $p$  (аналогично для образца  $q$ ), а  $n$  — значение размерности.

Если для кластера  $j$  рассмотреть вектор  $p_j$ , определяемый центроидом (точкой, соответствующей усредненной характеристике размещения всех образцов в кластере), то для данных на рис. 1 решение о том, какому из кластеров принадлежит произвольный вектор  $x$  (изображаемый на плоскости в виде точки), определяется значением

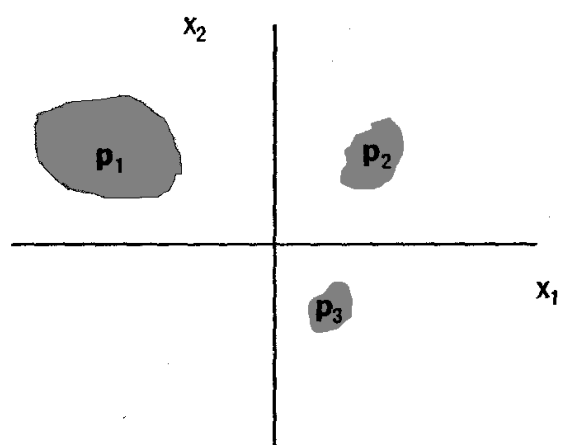
$$\text{index}(x) = \min d(p_j, x), \text{ для всех } j,$$

возвращающим индекс кластера с наименьшим квадратом евклидова расстояния до вектора  $x$ . Векторы  $p_j$  могут рассматриваться как прототипы кластеров, и эти прототипы могут служить для представления ключевых признаков кластера. Например, если необходимо разделить на группы баскетболистов и хоккеистов, несомненно, что в качестве характерного признака можно выбрать рост. Поэтому значения элементов, означающих рост в векторах-прототипах двух кластеров, должны существенно отличаться.

Алгоритм кластеризации представляет собой статистическую процедуру выделения групп из имеющегося набора данных. Один из самых простых подходов заключается в том, чтобы предположить существование определенного числа кластеров и произвольным образом выбрать координаты для каждого из прототипов. Затем каждый вектор из набора данных связывается с ближайшим к нему прототипом, и новыми прототипами становятся центроиды всех векторов, связанных с исходным прототипом. На рис. 2 показан случайный выбор прототипов, которые к концу обучения должны переместиться в центры кластеров, как показано на рис. 3.



*Рис. 2. Три случайных вектора, которые будут смещаться, вступая в роли прототипов для кластеров*



*Рис. 3. Каждый из прототипов переместился в точку, соответствующую центроиду кластера*

### Пример 1

Найдите квадрат евклидова расстояния между векторами:

$$p = [-2.3 \ 1.4], x = [4.5 \ 0.6].$$

#### Решение примера 1

$$d(p,x) = (-2.3 - 4.5)^2 + (1.4 - 0.6)^2 = 46.9.$$

### Пример 2

Найдите квадрат евклидова расстояния между векторами:

$$p = [0.4 \ 0.3 \ 1.1 \ 0.9], x = [0.6 \ 0.7 \ -0.5 \ 1.1].$$

#### Решение примера 2

$$d(p,x) = (0.4-0.6)^2 + (0.3-0.7)^2 + (1.1--0.5)^2 + (0.9-1.1)^2 = 2.8.$$

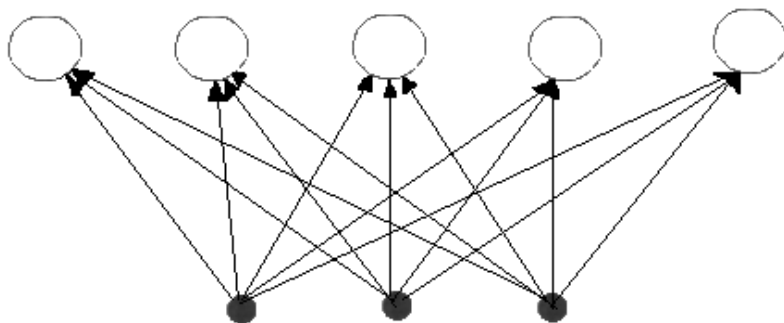
Прототипы можно рассматривать, как резюме экзаменуемого набора данных. Кластеры на рис. 1 имеют разные размеры, и самый большой из них имеет достаточно большую протяженность по обеим осям. Иногда бывает удобно представлять кластер несколькими прототипами, чтобы получить более детальную характеристику данных. Чтобы распознать кластеры, являющиеся частями большего кластера, необходимо знать положение всех прототипов друг относительно друга. Одной из проблем применения алгоритмов кластеризации является выбор оптимального числа кластеров. Если число кластеров выбрать слишком малым, могут быть упущены некоторые важные характеристики данных, а если кластеров окажется слишком много, то мы не получим никакой эффективной итоговой информации о данных (может даже случиться, что каждый образец создаст свой кластер). Можно сформулировать некоторые основные свойства идеального алгоритма кластеризации:

- автоматическое определение числа прототипов;
- сравнение прототипов;
- представление характерных признаков прототипа.

На практике первым из указанных свойств не обладает ни один из известных алгоритмов кластеризации. Нейронная сеть с обучением без управления, выполняющая кластеризацию, представляет собой самоорганизующуюся карту признаков, которую в начале 80-х годов предложил Кохонен (Kohonen).

## Самоорганизующаяся карта признаков

Самоорганизующаяся карта признаков (сеть SOFM — Self-Organizing Feature Map) имеет набор входных элементов, число которых соответствует размерности учебных векторов, и набор выходных элементов, которые служат в качестве



прототипов. Базовая архитектура сети SOFM показана на рис. 4.

*Рис. 4. Эта сеть имеет три входных и пять кластерных каждый элемент входного слоя связан с каждым элементом кластерного слоя*

Например, для данных, показанных на рис. 1, потребуется сеть с двумя входными и по крайней мере тремя выходными элементами, представляющими три кластера.

Входные элементы предназначены только для того, чтобы распределять данные входного вектора между выходными элементами сети. Выходные элементы называются кластерными элементами. Так как число входных элементов соответствует размерности вводимых векторов, а каждый входной элемент связан со всеми кластерными элементами, общее число влияющих на кластерный элемент весовых значений тоже оказывается равным размерности входных векторов. Часто удобно интерпретировать весовые значения кластерного элемента как значения координат, описывающих позицию кластера в пространстве входных данных. На рис. 5 показано, как связываются весовые значения с пространством входных данных.

Кластерные элементы размещаются в виде одно- или двумерного массива. В ходе обучения все элементы могут рассматриваться как претенденты на награды в виде учебных векторов. Когда на конкурс выставляется какой-либо учебный вектор, вычисляются расстояния от него до всех кластерных элемен-

тов, и элемент, который находится к данному учебному вектору ближе всех, объявляется элементом-победителем.

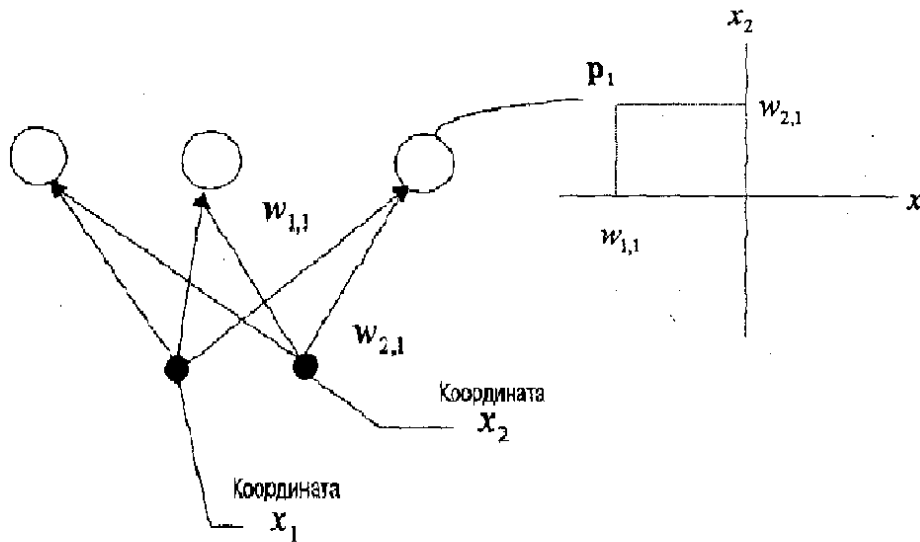


Рис. 5.

Для элемента-победителя выполняется корректировка весовых значений так, чтобы этот кластерный, элемент стал к учебному вектору еще ближе. Обычно корректировка весовых значений выполняется и для элементов, близких к элементу-победителю. Весовые значения элемента подлежат обновлению, если элемент лежит внутри круга заданного радиуса с центром в элементе-победителе. В ходе обучения радиус обычно постепенно уменьшается. Норма обучения ограничивает величину, на которую кластерный элемент может передвинуться по направлению к учебному вектору, и, подобно радиусу, норма обучения тоже со временем постепенно уменьшается. От топологии зависит только то, какие элементы должны обновляться для данного конкретного радиуса.

Обычно число кластерных элементов выбирают меньшим, чем число учебных образцов, поскольку целью является получение упрощенной характеристики данных. К концу обучения кластерные элементы обеспечивают "информационную сводку" по пространству входных образцов. Кластерные элементы выступают в роли карты признаков пространства входных данных.

### Алгоритм

В следующем алгоритме  $\eta$  обозначает норму обучения, а  $n$  - шаг во времени.

инициализировать весовые значения случайными значениями  
выполнять, пока not HALT

для каждого входного вектора

для каждого кластерного элемента вычислить расстояние до учебного вектора:

$$d_j = \sum_i (w_{ij} - x_i)^2$$

найти элемент  $j$ , для которого расстояние минимально

для элементов из круга заданного радиуса обновить весовые векторы по формуле

$$W_{ij}(n+1) = w_{ij}(n) + \eta(n)[x_i - w_{ij}(n)]$$

проверить, требуется ли обновление нормы обучения или радиуса

проверить HALT

Учебные векторы выбираются из набора векторов случайным образом.

Условие HALT (останов) выполняется тогда, когда величины изменения весов для всех кластерных элементов становятся очень маленькими - при этом условии учебные векторы должны попадать в одни и те же зоны карты при переходе от одной эпохи к следующей.

Норма обучения со временем меняется. Она может, например, сначала иметь значение 0.9, а затем уменьшаться линейно до некоторого фиксированного малого значения (скажем, 0.01), после чего оставаться неизменной. Радиус обычно выбирается достаточно большим, чтобы сначала обновлялись все элементы. Радиус тоже со временем уменьшается, и в конце, как правило, должны обновляться только несколько соседствующих с элементом-победителем элементов или вообще только сам этот элемент. Норма обучения тоже может зависеть от того, насколько близко размещается обновляемый элемент к элементу-победителю.

### Пример 3

1. Сколько эпох потребуется для того, чтобы значение  $\eta$  уменьшилось до 0.1, при условии использования следующего правила обновления:

$$\eta(0) = 0.09, \quad \eta(n+1) = \eta(n) - 0.001$$

2. Сеть SOFM имеет вид двумерной сетки размером  $10 \times 10$ , а радиус изначально задан равным 6. Выясните, как много элементов должны будут об-

новляться после 1000 эпох, если элемент-победитель размещен в правом нижнем углу сетки, а радиус меняется по правилу:

$$r = r - 1, \text{ если номер\_текущей\_эпохи mod } 200 = 0.$$

Предполагается, что нумерация эпох начинается с 1.

### Решение 3

$$\begin{aligned} 1. \quad & 0.1 = 0.9 - n * 0.001, \\ & \therefore n = (0.9 - 0.1) / 0.001 = 800. \end{aligned}$$

2. Если предположить, что счет эпох начинается с 1, радиус будет уменьшаться на 1 после каждых 200 эпох. После 1000 эпох радиус окажется равным 1. Число изменяемых элементов будет равно четырем, включая элемент-победитель.

### Обучение сети SOFM

Карта признаков проходит два этапа обучения. На первом этапе элементы упорядочиваются так, чтобы отражать пространство входных элементов, а на втором этапе происходит уточнение их позиций. Как правило, процесс представляется визуально путем использования двумерных данных и построения соответствующей поверхности.

### Пример 4

Для обучения сети SOFM с тремя входными и двумя кластерными элементами используются четыре учебных вектора:

$$[0.8 \ 0.7 \ 0.4], \quad [0.6 \ 0.9 \ 0.9], \quad [0.3 \ 0.4 \ 0.1], \quad [0.1 \ 0.1 \ 0.3]$$

и начальные весовые значения

$$\begin{bmatrix} 0,5 & 0,4 \\ 0,6 & 0,2 \\ 0,8 & 0,5 \end{bmatrix}$$

Начальный радиус выбирается равным 0, а норма обучения  $\eta$  – равной 0.5. Вычислите изменения весовых значений в ходе первого цикла обработки данных, рассматривая учебные векторы в указанном порядке.

### Решение примера 4

Рассматривая входной вектор 1, для кластерного элемента 1 получаем

$$d_1 = (0.5-0.8)^2 + (0.6-0.7)^2 + (0.8-0.4)^2 = 0.26,$$

а для кластерного элемента 2 –

$$d_2 = (0.4 - 0.8)^2 + (0.2 - 0.7)^2 + (0.5 - 0.4)^2 = 0.42.$$

Элемент 1 оказывается ближе, поэтому

$$w_{ij}(n+1) = w_{ij}(n) + 0.5[x_i - w_{ij}(n)].$$

Новыми весовыми значениями являются

$$\begin{bmatrix} 0,65 & 0,4 \\ 0,65 & 0,2 \\ 0,65 & 0,5 \end{bmatrix}$$

Рассматривая входной вектор 2, для кластерного элемента 1 получим

$$d_1 = (0.65 - 0.6)^2 + (0.65 - 0.9)^2 + (0.60 - 0.9)^2 = 0.155,$$

а для кластерного элемента 2 –

$$d_2 = (0.4 - 0.6)^2 + (0.2 - 0.9)^2 + (0.5 - 0.9)^2 = 0.69 .$$

Элемент 2 оказывается ближе, и новыми весовыми значениями будут

$$\begin{bmatrix} 0,625 & 0,400 \\ 0,775 & 0,200 \\ 0,750 & 0,500 \end{bmatrix}$$

Рассматривая входной вектор 3, для кластерного элемента 1 получим

$$d_1 = (0.625 - 0.3)^2 + (0.775 - 0.4)^2 + (0.75 - 0.1)^2 = 0.67,$$

а для кластерного элемента 2 –

$$d_2 = (0.4 - 0.3)^2 + (0.2-0.4)^2 + (0.5-0.1)^2 = 0.21.$$

Элемент 1 оказывается ближе, и новыми весовыми значениями будут

$$\begin{bmatrix} 0,625 & 0,350 \\ 0,775 & 0,300 \\ 0,750 & 0,300 \end{bmatrix}$$

Рассматривая входной вектор 4, для кластерного элемента 1 получим

$$d_1 = (0.625 - 0.1)^2 + (0.775-0.1)^2 + (0.75-0.3)^2 = 0.93,$$

а для кластерного элемента 2 —

$$d_2 = (0.35-0.10)^2 + (0.30-0.10)^2 + (0.30-0.30)^2 = 0.10.$$



Элемент 2 оказывается ближе, и новыми значениями весов будут

$$\begin{bmatrix} 0,625 & 0,225 \\ 0,775 & 0,200 \\ 0,750 & 0,300 \end{bmatrix}.$$

### Дополнительные сведения о сети SOFM

В определенных случаях в качестве меры сходства векторов можно использовать угол между ними. Взгляните на рис. 6. Вектор  $a$  ближе к прототипу  $p_1$  в смысле евклидова расстояния, но ближе к прототипу  $p_2$ , если в качестве меры сходства выбрать значение угла. Произведение (называемое также скалярным произведением) векторов

$$v = [v_1 \ v_2 \ \dots \ v_n] \text{ и } w = [w_1 \ w_2 \ \dots \ w_n]$$

определяется формулой

$$v \cdot w = [v_1 w_1 \quad v_2 w_2 \quad v_n w_n].$$

Угол между ненулевыми векторами  $v$  и  $w$  равен

$$\cos^{-1} \left( \frac{v \cdot w}{\|v\| \|w\|} \right).$$

Норма или модуль вектора вычисляется по формуле  $\|v\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$ . Вектор можно нормализовать путем деления каждого элемента вектора на норму вектора. Если все векторы нормализованы, то индекс прототипа-победителя для входного образца задается условием:

$$\text{index}(x) = \max \{p_j * x\}, \text{ для всех } j.$$

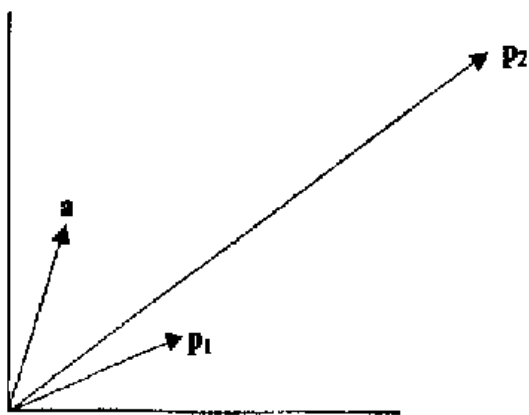


Рис. 6. Три вектора, используемые в мере 5.

Обучение сети SOFM, в которой в качестве меры сходства используется скалярное произведение векторов, проводится также, как было описано выше, но формула обновления весов теперь должна быть следующей:

$$w_j(n+1) = \frac{w_j(n) + \eta x}{\|w_j(n) + \eta x\|}.$$

Таким образом, для элемента-победителя к его весовому вектору добавляется часть этого вектора, а затем полученный в результате вектор нормализуется.

### Пример 5

Пусть векторам на рис. 6 являются

$$a=[1 \ 4], p_1=[2 \ 1], p_2=[6 \ 6].$$

- (а) Используя скалярное произведение, покажите, что  $a$  ближе к  $p_2$ , чем к  $p_1$ .
- (б) Если  $p$  оказывается прототипом-победителем в некоторой сети SOFM, то покажите, как должен двигаться вектор  $p$  в предположении, что вектор  $a$  предлагается на рассмотрение сети два раза подряд (т.е. в случае, когда нет других учебных образцов). Считайте, что  $\eta=1$ .

### Решение примера 5

(а) Нормализованные скалярные произведения равны:

$$ap_1^T = \frac{(1*2) + (4*1)}{\sqrt{1^2 + 4^2} \sqrt{2^2 + 1^2}} = 0.651,$$

$$ap_2^T = \frac{(1*6) + (4*6)}{\sqrt{1^2 + 4^2} \sqrt{6^2 + 6^2}} = 0.857.$$

Так что  $p_2$  оказывается ближе. Обратите внимание на то, что  $ap_j^T$  используется здесь для обозначения нормализованного произведения, где верхний индекс означает транспонирование вектора.

(б) Нормализованными векторами являются

$$a = \left[ \frac{1}{\sqrt{17}}, \frac{4}{\sqrt{17}} \right],$$

$$p_2 = \left[ \frac{6}{\sqrt{72}}, \frac{6}{\sqrt{72}} \right].$$

Для первого представления имеем:

$$p_2(n+1) = \frac{[6/\sqrt{72} + 1/\sqrt{17}, 6/\sqrt{72} + 4/\sqrt{17}]}{\sqrt{(6/\sqrt{72} + 1/\sqrt{17})^2 + (6/\sqrt{72} + 4/\sqrt{17})^2}} = [0.493, 0.870].$$

Для второго представления имеем:

$$p_2(n+2) = \frac{[0.493 + 1/\sqrt{17}, 0.870 + 4/\sqrt{17}]}{\sqrt{(0.493 + 1/\sqrt{17})^2 + (0.870 + 4/\sqrt{17})^2}} = [0.371, 0.928].$$

### Упражнения

1. Векторы  $x$ ,  $p_1$  и  $p_2$  являются следующими:

$$x = [0.2 \ -1.4 \ 2.3], \quad p_1 = [0.6 \ -4.0 \ 7.0], \quad p_2 = [0.1 \ -1.0 \ 2.2].$$

(а) К какому из прототипов оказывается ближе всего вектор  $x$  в смысле евклидова расстояния?

(б) К какому из прототипов оказывается ближе всего вектор  $x$  в смысле скалярного произведения?

(в) Скорректируйте весовой вектор прототипа-победителя из п. (а) в соответствии с алгоритмом обучения SOFM при норма обучения 0.8.

(г) Скорректируйте весовой вектор прототипа-победителя из п. (а) в соответствии с алгоритмом обучения SOFM для случая использования скалярного произведения при норме обучения 0.8.

2. Повторите вычисления упражнения 1 для следующих векторов:

$$x = [0.2 \ -1.4 \ -0.3 \ 0.8], \quad p_1 = [0.3 \ -3.0 \ 1.0 \ 0.2], \quad p_2 = [0.4 \ -1.4 \ -2.0 \ 3.0].$$

3. Норма обучения в сети SOFM уменьшается в течение первых 1000 итераций по закону  $\eta(n) = 0.15(1 - \frac{n}{1000})$ , где  $n$  обозначает номер итерации.

(а) Сколько итераций будет выполнено прежде, чем норма обучения уменьшится до значения 0.003?

(б) Почему указанный закон не является хорошим выбором?

4. Предположим, что точки  $\{(-1, 1), (-1, -1), (1, -1)\}$  принадлежат классу А, а точки  $\{(-2, -2), (1, 1), (2, 2), (4, 1)\}$  — классу В.

(а) Докажите, что эти классы не являются линейно отделимыми.

(б) Предположив, что выход элементов сети задается условием

$$\text{выход} = \begin{cases} 1, & \text{если } \_ \text{комбинированный\_ввод} \geq 0 \\ 0, & \text{если } \_ \text{комбинированный\_ввод} < 0 \end{cases}$$

покажите, что определенный ниже матрицей  $w_1$  первый слой весовых значений в сети с тремя слоями преобразует проблему в линейную (первая строка матрицы  $w_1$ ) определяет весовые коэффициенты смещения):

$$W_1 = \begin{bmatrix} 1 & -6 \\ -2 & -2 \\ -1 & -3 \end{bmatrix}.$$

(в) Определите значения второго слоя весов так, чтобы сеть правильно классифицировала указанные выше образцы. Предположите, что сеть имеет один выходной элемент.

5. Точки  $\{(4,-1), (8,-2), (1,1), (3,6)\}$  принадлежат классу А, а точки  $\{(-8,4), (-2,-3), (-1,-1), (2,-9)\}$  — классу В. Постройте минимальную сеть, правильно классифицирующую эти точки.
6. На рис. 7 показаны два класса, А и В, образующие треугольные области. Третий класс определяется как пересечение классов А и В. Предложите архитектуру сети с обратным распространением ошибок, решающей такую задачу классификации. Объясните, почему вы выбрали именно такую архитектуру.

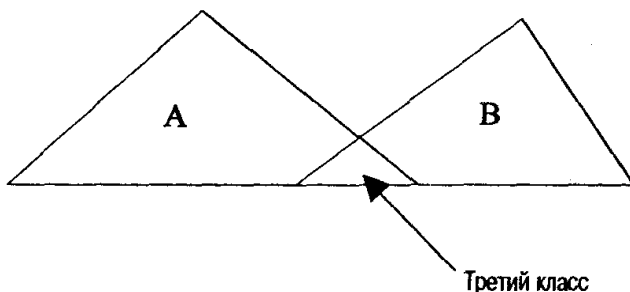


Рис. 7. Две треугольные области и их пересечение

7. На рис. 8 показаны два кластера данных (на рисунке они закрашены). Не закрашенная область в верхнем левом квадранте содержит несколько прототипов (единичных весовых векторов), выступающих в качестве

начальных состояний для сети SOFM. Принимая радиус равным нулю, объясните, что может происходить с картой признаков, если начать обучение сети с этими данными.

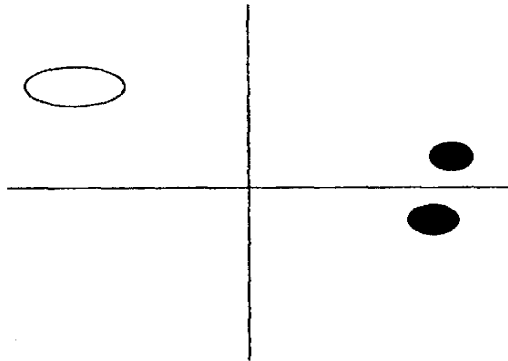


Рис. 8. Два кластера данных и прототипы для сети SOFM

8. Сеть MAXNET представляет собой конкурентную нейронную сеть, которая может использоваться для нахождения элемента сети, ввод которого оказывается максимальным. Каждый элемент соединен с самим собой, а также со всеми другими элементами двунаправленными взвешенными связями. Все весовые значения устанавливаются равными между собой, кроме веса автосвязи, который устанавливается равным 1:

$$w_{ij} = \begin{cases} 1, & \text{если } i = j \\ -\omega, & \text{если } i \neq j \end{cases}$$

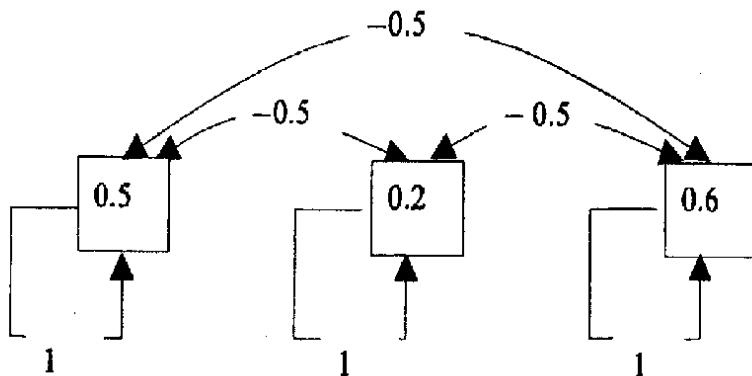
где  $0 < \omega < 1/N$ , а N равно числу элементов сети.

Значение активности элемента устанавливается равным вводу, если ввод оказывается больше нуля, а иначе значение активности устанавливается равным нулю. Элемент получает взвешенный сигнал от всех других элементов и от себя самого. Элемент не изменяет свою активность до завершения итерации. Обновляются все элементы, и обновление происходит до тех пор, пока останется не более одного элемента с ненулевой активностью.

На рис. 9 показаны значения активности трех элементов при условии, что значения весов были инициализированы в соответствии с вышеуказанными правилами.

Для первой итерации вводом первого элемента будет

$$0.5 + 0.2 * -0.5 + 0.6 * -0.5 = 0.1.$$



*Рис. 9. Значения активности трех элементов*

Новым значением активности будет 0.1, поскольку ввод оказывается больше нуля. До конца итерации активность элемента будет оставаться равной 0.5. Ввод для второго элемента равен  $0.2 + 0.5 \times -0.5 + 0.6 \times -0.5 = -0.35$ . Новым значением активности будет 0, так как ввод оказывается меньше нуля. До конца первой итерации активность элемента будет оставаться равной 0.2. Ввод для третьего элемента равен  $0.6 + 0.5 \times -0.5 + 0.2 \times -0.5 = 0.25$ . Новым значением активности будет 0.25, поскольку ввод оказывается больше нуля. В данный момент все элементы могут изменить значения активности.

Для второй итерации вводом первого элемента будет

$$0.1 + 0 + 0.25 \times -0.5 = -0.025.$$

Поэтому новым значением активности в конце итерации будет ноль. Ввод для второго элемента должен быть меньше нуля. Ввод для третьего элемента равен  $0.25 + 0.1 \times -0.5 + 0 = 0.2$ . Единственным элементом с ненулевой активностью оказывается третий элемент. Поэтому процесс завершится, и победителем будет третий элемент.

Повторите вышеприведенные рассуждения для значений активности элементов, равных 0.7, 0.6 и 0.3, соответственно.

### **Практическая работа № 5**

**Тема: Ассоциация образов**

Рассмотрим случай запоминаемых пар образцов. Идея заключается в том, чтобы выбрать нужный образец из памяти, даже если у нас нет всей необходимой информации для начала поиска сохраненного образца. Например, вы хотите найти книгу в библиотеке, но не помните ее названия. При этом если вы знаете имя автора и описание того, чему книга посвящена, этого уже достаточно (с большой долей уверенности!), чтобы найти ассоциируемый с этой информацией объект.

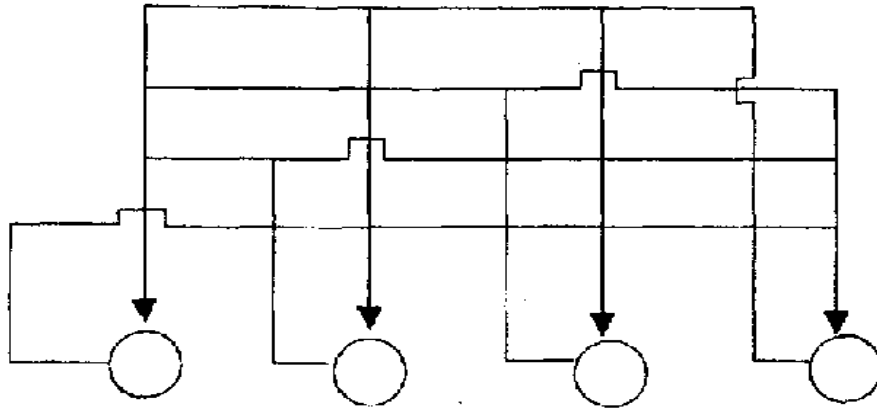
Когда сохраняемая в памяти пара ассоциируемых образцов создается одинаковыми образцами, память называется автоассоциативной, а если образцы являются разными, то память называется гетероассоциативной. В этой главе будут рассмотрены три модели нейронных сетей для автоассоциации образцов.

### *Дискретная сеть Хопфилда*

•Сеть Хопфилда (Hopfield) является автоассоциативной сетью, ведущей себя подобно памяти, которая может вспомнить сохраненный образец даже по подсказке (в виде вводимых данных), представляющей собой искаженную помехами версию нужного образца. Например, сеть может сохранить набор изображений букв, а когда сети будет представлена искаженная версия сохраненного символа, сеть должна оказаться способной найти истинный экземпляр.

Сеть Хопфилда является рекуррентной в том смысле, что для каждого входного образца выход сети повторно используется в качестве ввода до тех пор, пока не будет достигнуто устойчивое состояние. Пример сети Хопфилда показан на рис. 1. Удобно считать, что сеть Хопфилда не имеет входных элементов, так как входной вектор просто определяет начальные значения активности элементов. Например, если ввод является двоичным, то входной вектор  $[1 \ 1 \ 0 \ 1]$  означает, что значения активности для элементов  $\{1, 2, 4\}$  будут равны 1, а для элемента  $\{3\}$  активность будет равна 0. Элемент обновляется тогда, когда все элементы передадут свои значения активности по имеющимся взвешенным связям, после чего вычисляется сумма произведений (т.е. берется скалярное произведение). Значение активности элемента получается на основе ис-

пользования некоторого правила активизации. Каждый элемент сети Хопфилда имеет состояние, характеризующееся значением активности, которое должен посылать данный элемент другим элементам, а состояние сети в любой момент времени задается вектором состояний всех ее элементов.



*Рис. 1. Сеть Хопфилда с четырьмя элементами. Для каждого элемента входного вектора имеется свой элемент сети. Элементы сети связаны со всеми остальными ее элементами, но не сами с собой. Связи являются двунаправленными*

В качестве входных данных сети Хопфилда можно использовать двоичные, но здесь мы будем использовать +1 для обозначения состояния "включено" и -1 — для состояния "выключено". Комбинированный ввод элемента вычисляется по формуле:

$$net_i = \sum_{i=1}^n s_i w_{ij} ,$$

где  $s_i$  обозначает состояние элемента с номером  $i$ . Когда элемент обновляется, его состояние изменяется в соответствии с правилом

$$s_j = \begin{cases} +1, & \text{если } net_j > 0 \\ -1, & \text{если } net_j < 0 \end{cases}$$

Эта зависимость называется сигнум-функцией и в более краткой форме она записывается в виде

$$s_j = \text{sgn}(net_j).$$



Если комбинированный ввод оказывается равным нулю, то элемент остается в состоянии, в котором он пребывал перед обновлением.

Сеть работает очень просто. Входной вектор задает начальные состояния всех элементов. Элемент для обновления выбирается случайным образом. Выбранный элемент получает взвешенные сигналы от всех остальных элементов и изменяет свое состояние. Выбирается другой элемент, и процесс повторяется. Сеть достигает предела, когда ни один из ее элементов, будучи выбранным для обновления, не меняет своего состояния.

Весовые значения для сети Хопфилда определяются непосредственно из учебных данных без необходимости проведения обучения в более привычном смысле. Сеть Хопфилда ведет себя как память, и процедура сохранения отдельного вектора представляет собой вычисление прямого произведения вектора с ним самим. В результате этой процедуры создается матрица, задающая весовые значения для сети Хопфилда, в которой все диагональные элементы должны быть установлены равными нулю (поскольку диагональные элементы задают автосвязи элементов, а элементы сами с собой не связаны). Таким образом, весовая матрица, соответствующая сохранению вектора  $x$ , задается формулой  $W = x^T x$ .

**Пример 1.** Найдите набор весовых значений сети Хопфилда, соответствующий сохранению образца  $[1 \ -1 \ 1 \ 1]$ .

**Решение 1.**

$$\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} [1 \ -1 \ 1 \ 1] = \begin{bmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}$$

Поэтому весовыми значениями будут

$$W = \begin{bmatrix} 0 & -1 & 1 & 1 \\ -1 & 0 & -1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 1 & 0 \end{bmatrix}.$$

Первый столбец представляет весовые значения, связанные с первым элементом, столбец 2 представляет весовые значения, связанные со вторым элементом, и т.д. Если сети будет предложен образец  $[1 -1 1 1]$ , то все элементы после обновления останутся в том же состоянии. Данные подсказки определяют начальные состояния всех элементов, так что в нашем случае второй элемент должен сначала находиться в состоянии  $-1$ , а все остальные — в состоянии  $1$ . Первый элемент обновляется с помощью умножения вектора подсказ на первый столбец матрицы весов:

$$[1 \ -1 \ 1 \ 1] \begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \end{bmatrix} = 3, \text{sgn}(3)=1.$$

Так что первый элемент останется в том же состоянии. Точно также при обновлении оставались бы неизменными и состояния всех остальных элементов.

### **Пример 2.**

Найдите устойчивое состояние сети Хопфилда из примера 1 при условии, что входным образцом является  $[-1 -1 1 1]$ .

### **Решение 2.**

Элементы должны обновляться в случайном порядке. Для иллюстрации будем обновлять элементы в порядке 3, 4, 1, 2. Сначала рассмотрим элемент 3:

$$[-1 \ -1 \ 1 \ 1] \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \end{bmatrix} = 1, \text{sgn}(1)=1.$$

Таким образом, элемент 3 остается в том же состоянии. Следующим является элемент 4:

$$[-1 \ -1 \ 1 \ 1] \begin{bmatrix} 1 \\ -1 \\ 1 \\ 0 \end{bmatrix} = 1, \text{sgn}(1)=1$$

Так что элемент 4 остается в том же состоянии. Теперь элемент 1:

$$[-1 \ -1 \ 1 \ 1] \begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \end{bmatrix} = 3, \text{sgn}(3)=1$$

Так что элемент 1 изменит свое состояние с -1 на 1. Наконец, элемент 2:

$$[1 \ -1 \ 1 \ 1] \begin{bmatrix} -1 \\ 0 \\ -1 \\ -1 \end{bmatrix} = 3, \text{sgn}(-3)=-1$$

Так что элемент 2 останется в том же состоянии. Мы видим, что выявился ранее сохраненный вектор, характеризующий устойчивое состояние сети. Чтобы убедиться в том, что это состояние на самом деле является устойчивым, необходимо проверить, что в результате обновления ни один из элементов действительно не изменит своего состояния.

Процедура сохранения нескольких образцов в сети Хопфилда тоже проста: прямое произведение вычисляется для каждого вектора, и все полученные таким образом весовые матрицы складываются.

### Пример 3.

Определите весовую матрицу сети Хопфилда, соответствующую сохранению следующих двух векторов  $[-1 \ 1 \ -1]$ ,  $[1 \ -1 \ 1]$ .

### Решение 3.

Соответствующей весовой матрицей является матрица

$$W = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} [-1 \ 1 \ -1] + \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} [1 \ -1 \ 1] = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{bmatrix}$$

Диагональные элементы были обнулены.

### *Двунаправленная ассоциативная память*

Сетью, имеющей много общего с сетью Хопфилда, является двунаправленная ассоциативная память (сеть ВАРМ — Bidirectional Associate Memory), предложенная Коско (см. [Kosko, 1988]). Сеть ВАРМ является гетероассоциативной рекуррентной сетью. Сеть сохраняет пары образцов и может восстановить образец, когда ассоциированный с ним образец предлагается ей в качестве

подсказки. В этой сети два слоя элементов — по одному для каждого из образцов пары — и оба слоя соединяются двунаправленными связями (т.е. активность может передаваться по связям в обоих направлениях) (рис. 2).

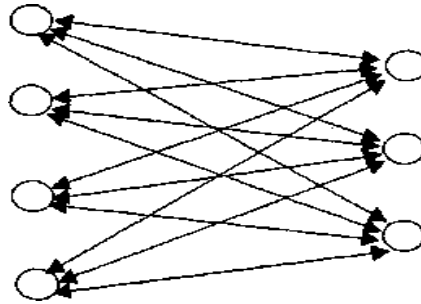


Рис. 2. Двунаправленная ассоциативная память. Элементы слева представляют образцы размерности 4, а элементы справа - ассоциированные с ними образцы размерности 3

Здесь мы рассмотрим только дискретную биполярную сеть ВАРМ, но можно рассмотреть и непрерывные значения. Чтобы сохранить образец  $s$  и ассоциируемый с ним образец  $t$ , рассматривается прямое произведение, определяющее весовые значения. Процедура точно такая же, как и в сети Хопфилда, но теперь матрица уже не обязана быть квадратной, а диагональные элементы не обнуляются. Весовой матрицей для одной пары будет матрица

$$W = s^T t.$$

Чтобы сохранить несколько пар, все соответствующие произведения, определяющие весовые значения, складываются, точно так же, как это делается для сети Хопфилда.

Процедура нахождения в памяти элемента подобна соответствующей процедуре сети Хопфилда.

$$t_j = f(net_j) = \begin{cases} 1, \text{если } net_j > \theta_j \\ t_j, \text{если } net_j = \theta_j \\ -1, \text{если } net_j < \theta_j \end{cases}$$

$$s_i = f(net_i) = \begin{cases} 1, \text{если } net_i > \theta_i \\ s_i, \text{если } net_i = \theta_i \\ -1, \text{если } net_i < \theta_i \end{cases}$$

Все элементы сети сначала имеют нулевые значения активности. Обратите внимание на то, что распространение может начаться с любого уровня, так как  $s$  может использоваться для вызова  $t$ , и, наоборот,  $-t$  может использоваться для вызова  $s$ .

#### Пример 4.

1. На рис. 3 показаны три образца (изображения цифр 1, 2 и 3). Эти три образца должны быть сохранены биполярной сетью ВМ. Ассоциируемыми с ними образцами являются трехбитовые двоичные числа (конвертированные в биполярную форму). Ассоциированные образцы представлены в табл. 1. Предложите весовые значения для этой сети.

2. Покажите, что каждая ассоциация может быть вызвана из памяти.

Таблица 1. Цифры {1,2,3}, ассоциированные с биполярными образцами

Образец	Ассоциированный образец
1	-1 -1 1
2	-1 1 -1
3	-1 1 1

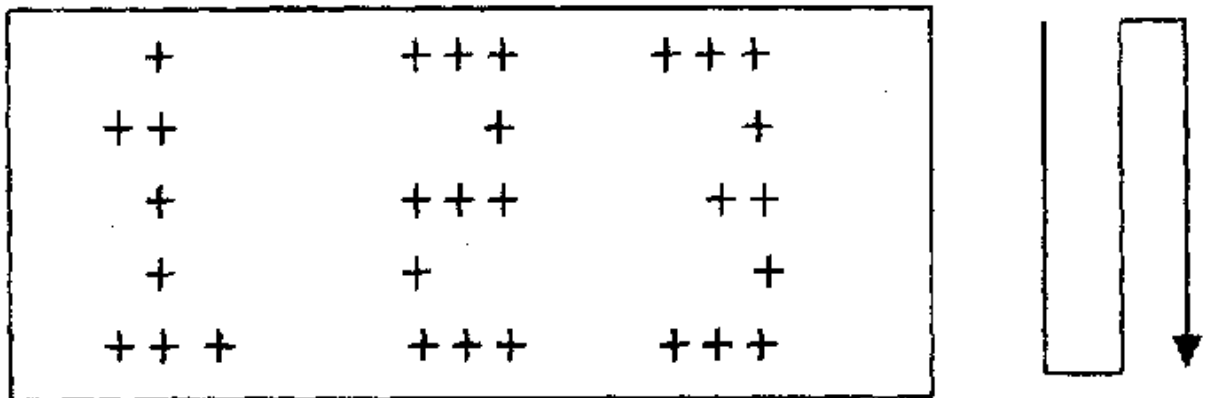


Рис. 4.5. Каждая цифра изображена на сетке 5x3. Пробел кодируется значением  $-1$ , а знак '+' - значением  $+1$ . Цифры представляются линейным массивом значений, получаемым при движении по сетке сверху вниз и слева направо

#### Решение 4.

1.

$$W = \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 \end{bmatrix} \quad \therefore W = \begin{bmatrix} -1 & 3 & -1 \\ 1 & -3 & 1 \\ 1 & 1 & -3 \\ 1 & 1 & -3 \\ -3 & 1 & 1 \\ -3 & 1 & 1 \\ 1 & -3 & 1 \\ -3 & 1 & 1 \\ 1 & -3 & 1 \\ -3 & 1 & 1 \\ -1 & 3 & -1 \\ -1 & 3 & -1 \\ -1 & 3 & -1 \\ 1 & 1 & 1 \\ -3 & 1 & 1 \end{bmatrix}$$

2. Сначала в качестве входного рассмотрим образец, представляющий оцифрованное изображение цифры "2":

комбинированный ввод для слоя j

$$= \begin{bmatrix} 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -1 \\ 1 & -3 & 1 \\ 1 & 1 & -3 \\ 1 & 1 & -3 \\ -3 & 1 & 1 \\ -3 & 1 & 1 \\ 1 & -3 & 1 \\ -3 & 1 & 1 \\ -1 & 3 & -1 \\ -1 & 3 & -1 \\ -1 & 3 & -1 \\ 1 & 1 & 1 \\ -3 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -21 & 27 & -9 \end{bmatrix}$$

С учетом порогового значения получаем  $[-1 \ 1 \ -1]$ , что и будет ассоциируемым образцом.

Образец  $[-1 \ 1 \ -1]$  можно использовать в качестве входного:

комбинированный ввод для слоя j

$$= [-1 \ 1 \ -1] \begin{bmatrix} -1 & 3 & -1 \\ 1 & -3 & 1 \\ 1 & 1 & -3 \\ 1 & 1 & -3 \\ -3 & 1 & 1 \\ -3 & 1 & 1 \\ 1 & -3 & 1 \\ -3 & 1 & 1 \\ 1 & -3 & 1 \\ -3 & 1 & 1 \\ -1 & 3 & -1 \\ -1 & 3 & -1 \\ -1 & 3 & -1 \\ 1 & 1 & 1 \\ -3 & 1 & 1 \end{bmatrix}^T = \begin{bmatrix} 5 \\ -5 \\ 3 \\ 3 \\ 3 \\ 3 \\ -5 \\ 3 \\ -5 \\ 3 \\ 5 \\ 5 \\ 5 \\ -1 \\ 3 \end{bmatrix}^T .$$

Оригинальный соответствующий изображению образец воссоздается после прохождения полученного вектора через функцию активности.

Если процесс повторить для других векторов, будет видно, что вызываются все ассоциации.

## **9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ**

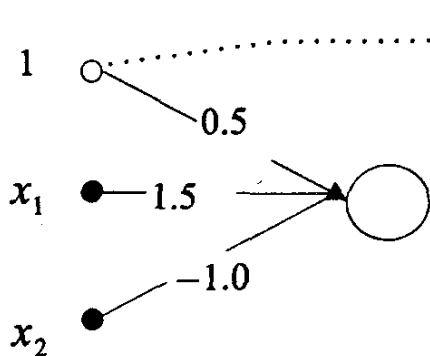
1. Межсессионная аттестация студентов проводится дважды в семестр на 7 и 13 неделях 5-го семестра.
2. Аттестационная оценка складывается из оценок, полученных по результатам промежуточного тестирования.
3. Организация аттестации студентов, проводится в соответствии с положением АмГУ о курсовых экзаменах и зачетах.



## 10. ФОНД КОНТРОЛЬНЫХ ЗАДАНИЙ ДЛЯ ОЦЕНКИ КАЧЕСТВА ЗНАНИЙ ПО ДИСЦИПЛИНЕ

Задание № 1.

Вычислите комбинированный сетевой ввод для элемента на рисунке и со-



ответствующее выходное значение при использовании пороговой функции и входного вектора [0.7 2.5].

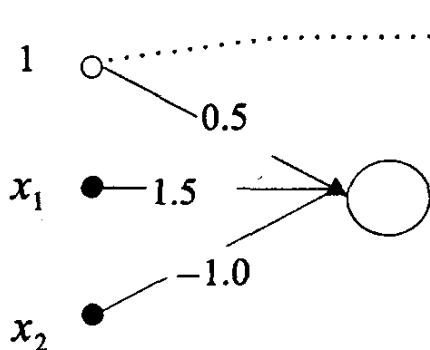
Задание №2.

Вычислите выходное значение, использовав в качестве функции активности сигмоидальную функцию. Входной вектор [0.7 2.5], весовая матрица

$$w = \begin{pmatrix} 0,5 \\ 1,5 \\ -1,0 \end{pmatrix}.$$

Задание № 3 .

Вычислите комбинированный ввод для сети с архитектурой, показанной на рисунке, но с набором весовых значений [-0.2 0.03 1.2] и входным векто-



ром [0.7 2.5].

Задание № 4.

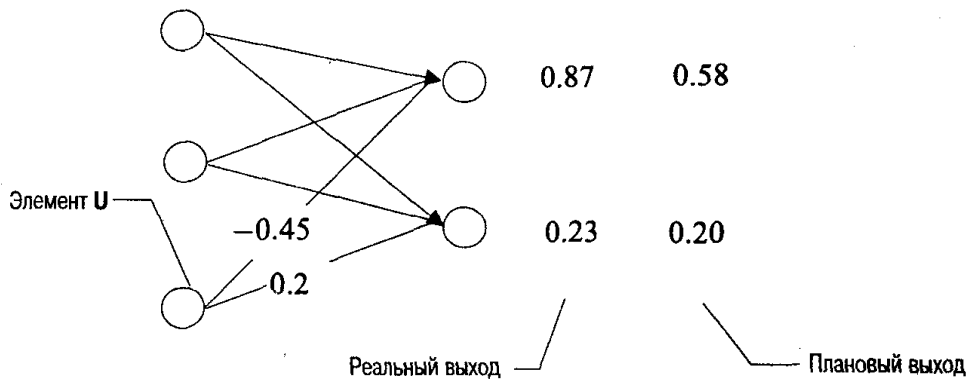
Представьте полностью прямой и обратный проходы в сети с прямой связью, использующей алгоритм обратного распространения ошибок, для входного образца [0.1 0.9] и целевого выходного значения 0.9 в предположении, что сеть имеет архитектуру 2-2-1 (т.е. два входных, два скрытых один выходной элемент) с

весовыми коэффициентами  $\begin{bmatrix} 0,1 & 0,1 \\ -0,2 & -0,1 \\ 0,1 & 0,3 \end{bmatrix}$  для первого слоя,  $\begin{bmatrix} 0,2 \\ 0,2 \\ 0,3 \end{bmatrix}$  для второго

слоя.

Задание № 5.

На рисунке



показаны скрытый и выходной слою сети с прямой связью. Вычислите ошибку для скрытого элемента U при условии, что значение его активности для обрабатываемого сетью образца равно 0.64.

Задание № 6.

Сеть типа 2-2-1 с радиальными базисными функциями используется для

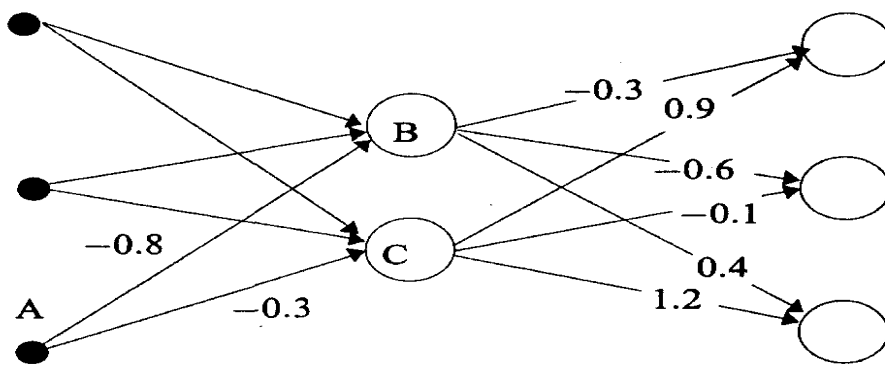
решения проблемы XOR. Первый слой весов задан матрицей  $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ .

Для каждого вводимого образца XOR вычислите значения активности для всех скрытых элементов, если функция активности имеет вид  $\sigma(net) = \exp[-net^2]$ , где  $net$  является евклидовой нормой.

Задание № 7.

На рисунке показана сеть с обратным распространением ошибок во время обработки учебного вектора  $[1.0 \ 0.9 \ 0.9]$ , для которого целевым выходным вектором является  $[0.1 \ 0.9 \ 0.1]$ . Пусть выходом элемента В является значение 0.6, а выходом элемента С — значение 0.8. Предположим, что функцией активности является сигмоид.

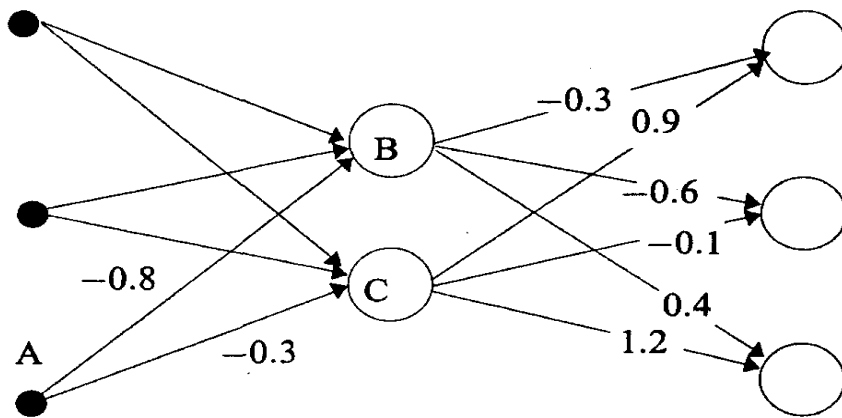
- (b) Вычислите фактический выходной вектор.
- (c) Вычислите значения ошибок для каждого выходного элемента.
- (d) Вычислите значения ошибок для каждого скрытого элемента.



Задание № 8.

На рисунке показана сеть с обратным распространением ошибок во время обработки учебного вектора  $[0.1 \ 0.5 \ 0.9]$ , для которого целевым выходным вектором является  $[0.1 \ 0.9 \ 0.1]$ . Пусть выходом элемента В является значение 0.6, а выходом элемента С — значение 0.8. Предположим, что функцией активности является сигмоид.

- (a) Вычислите фактический выходной вектор.
- (b) Вычислите значения ошибок для каждого выходного элемента.
- (c) Вычислите значения ошибок для каждого скрытого элемента.



Задание № 9.

Для обучения сети SOFM с тремя входными и двумя кластерными элементами используются четыре учебных вектора:  $[0.8 \ 0.7 \ 0.4]$ ,  $[0.6 \ 0.9 \ 0.9]$ ,  $[0.3 \ 0.4$

$0.1]$ ,  $[0.1 \ 0.1 \ 0.3]$  и начальные весовые значения  $\begin{bmatrix} 0,5 & 0,4 \\ 0,6 & 0,2 \\ 0,8 & 0,5 \end{bmatrix}$ .

Начальный радиус выбирается равным 0, а норма обучения  $\eta$  —ной 0.5. Вычислите изменения весовых значений в ходе первого цикла обработки данных, рассматривая учебные векторы в указанном порядке.

Задание № 10.

Пусть учебными векторам являются вектора  $a=[1 \ 4]$ ,  $p_1=[2 \ 1]$ ,  $p_2=[6 \ 6]$ .

(а) Используя скалярное произведение, покажите, что  $a$  ближе к  $p_2$ , чем к  $p_1$ .

(б) Если  $p$  оказывается прототипом-победителем в некоторой сети SOFM, то покажите, как должен двигаться вектор  $p$  в предположении, что вектор  $a$  предлагается на рассмотрение сети два раза подряд (т.е. в случае, когда нет других учебных образцов). Считайте, что  $\square=1$ .

Задание № 11.

Пусть учебными векторам являются вектора  $x = [0.2 \ -1.4 \ 2.3]$ ,  $p_1=[0.6 \ -4.0 \ 7.0]$ ,  $p_2=[0.1 \ -1.0 \ 2.2]$ .

(а) К какому из прототипов оказывается ближе всего вектор  $x$  в смысле евклидова расстояния?

(б) К какому из прототипов оказывается ближе всего вектор  $x$  в смысле скалярного произведения?

(в) Скорректируйте весовой вектор прототипа-победителя из п. (а) в соответствии с алгоритмом обучения SOFM при норма обучения 0.8.

Задание 12.

Пусть учебными векторам являются вектора  $x = [0.2 \ -1.4 \ -0.3 \ 0.8]$ ,  $p_1 = [0.3 \ -3.0 \ 1.0 \ 0.2]$ ,  $p_2 = [0.4 \ -1.4 \ -2.0 \ 3.0]$ .

(а) К какому из прототипов оказывается ближе всего вектор  $x$  в смысле евклидова расстояния?

(б) К какому из прототипов оказывается ближе всего вектор  $x$  в смысле скалярного произведения?

(в) Скорректируйте весовой вектор прототипа-победителя из п. (а) в соответствии с алгоритмом обучения SOFM при норма обучения 0.8.

Задание № 13.

Найдите набор весовых значений сети Хопфилда, соответствующий сохранению образца  $[1-111]$ .

Задание № 14.

Найдите устойчивое состояние сети Хопфилда при условии, что входным образцом является  $[-1-1 \ 1 \ 1]$ .

Задание № 15.

Определите весовую матрицу сети Хопфилда, соответствующую сохранению следующих двух векторов  $[-1 \ 1 \ -1]$ ,  $[1 \ -1 \ 1]$ .

Задание № 16.

Определите весовые значения сети Хопфилда, соответствующие сохранению образца  $[1 \ 1 \ 1 \ -1]$ .

Задание № 17.

Определите весовые значения сети Хопфилда для сохранения следующих образцов:  $[-1 \ 1 \ -1 \ -1 \ 1 \ -1 \ -1 \ -1]$ ,  $[-1 \ -1 \ 1 \ -1 \ -1 \ -1 \ 1 \ -1]$ .

Проверьте устойчивость сети при предоставлении ей на вход в качестве подсказок сохраненных образцов.

Задание № 18.

Проверьте устойчивость сети при предоставлении ей на вход в качестве подсказок следующих образцов:  $[-1 \ 1 \ -1 \ -1 \ 1 \ -1 \ -1 \ 1]$ ,  
 $[-1 \ 1 \ 1 \ -1 \ 1 \ -1 \ -1 \ -1]$ .

Задание № 19.

Определите весовые значения сети Хопфилда для сохранения следующих образцов:

$$\begin{aligned} &[-1 \ 1 \ -1 \ -1 \ 1 \ -1 \ -1 \ -1], \\ &[-1 \ -1 \ 1 \ -1 \ -1 \ -1 \ 1 \ -1], \\ &[1 \ -1 \ -1 \ 1 \ -1 \ 1 \ -1 \ -1]. \end{aligned}$$

Проверьте устойчивость сети при предоставлении ей на вход в качестве подсказки первого сохраненного образца.

## 11. КОНТРОЛЬНЫЕ ВОПРОСЫ К ЗАЧЕТУ

1. Свойства нейронных сетей.
2. Прикладные возможности нейронных сетей.
3. Топология нейронных сетей.
4. Биологический нейрон. Биологические основы функционирования нейрона.
5. Формальный нейрон. Модель МакКаллока-Питса. Весовая матрица.
6. Виды функций активации.
7. Перцептрон и методы его обучения.
8. Сигмоидальный нейрон.
9. Нейрон типа «адалайн».
10. Инстар и оутстар Гроссберга.
11. Нейроны типа WTA.
12. Модель нейрона Хебба.
13. Стахостическая модель нейрона.
14. Обучение нейронной сети.

15. Обучение по алгоритму обратного распространения ошибки.
16. Градиентные алгоритмы обучения сети. Основные положения.
17. Алгоритм наискорейшего спуска.
18. Алгоритм переменной метрики.
19. Алгоритм Левенберга – Марквардта.
20. Алгоритм сопряженных градиентов.
21. Подбор коэффициента обучения.
22. Подбор оптимальной архитектуры сети.
23. Многослойный перцептрон.
24. Кластеризация образцов. Основные положения.
25. Самоорганизующаяся карта признаков. Базовая архитектура сети.
26. Ассоциация образцов. Сеть Хопфилда.
27. Ассоциация образцов. Двухнаправленная ассоциативная память.
28. Автоассоциативное обратное распространение ошибок. Базовая архитектура сети.
29. Рекуррентные сети. Обратное распространение во времени.
30. Простая рекуррентная сеть.
31. Сеть Джордана.
32. Что такое метод модельной "закалки" и как он используется в нейронных сетях?
33. Принцип построения вероятностной нейронной сети. Свойства сети.
34. Машина Больцмана.
35. Вероятностные нейронные сети.
36. Модульная нейронная сеть.

## **12. КОМПЛЕКТ ЭКЗАМЕНАЦИОННЫХ БИЛЕТОВ**

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра ИУС

Утверждено на заседании кафедры  
« \_\_\_\_ » \_\_\_\_\_ 200\_г.  
Заведующий кафедрой  
Утверждаю \_\_\_\_\_

Факультет МиИ  
Курс II  
Дисциплина  
Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 1

1. Что такое нейронные сети?
2. Сеть Хопфилда.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры  
« \_\_\_\_ » \_\_\_\_\_ 200\_г.  
Заведующий кафедрой  
Утверждаю \_\_\_\_\_

Кафедра ИУС  
Факультет МиИ  
Курс II  
Дисциплина  
Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 2

1. Свойства биологических и искусственных нейронных сетей.
2. Двухнаправленная ассоциативная память.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры  
« \_\_\_\_ » \_\_\_\_\_ 200\_г.  
Заведующий кафедрой

Кафедра ИУС  
Факультет МиИ  
Курс II  
Дисциплина



Утверждаю \_\_\_\_\_

Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 3

1. Способы реализации нейросетей
2. Автоассоциативное обратное распространение ошибок. Базовая архитектура сети.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой

Утверждаю \_\_\_\_\_

Кафедра ИУС

Факультет МиИ

Курс II

Дисциплина

Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 4

1. Типы задач, решаемых нейронными сетями. Недостатки и ограничения нейронных сетей.
2. Рекуррентные сети. Базовая архитектура сети.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой

Кафедра ИУС

Факультет МиИ

Курс II

Дисциплина

Утверждаю \_\_\_\_\_

Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 5

1. Биологический нейрон. Структура, функции.
2. Обратное распространение во времени. Базовая архитектура рекуррентной сети с обратным распространением ошибок.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой

Утверждаю \_\_\_\_\_

Кафедра ИУС

Факультет МиИ

Курс II

Дисциплина

Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 6

1. Формальный нейрон. Ограниченность модели формального нейрона.
2. Кластеризация образцов. Сеть Кохонена.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой

Кафедра ИУС

Факультет МиИ

Курс II

Дисциплина

Утверждаю \_\_\_\_\_

Нейроинформатика

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 7

1. Виды функций активации.
2. Обучение слоя Кохонена.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой

Утверждаю \_\_\_\_\_

Кафедра ИУС

Факультет МИИ

Курс II

Дисциплина

Нейроинформатика

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 8

1. Структура связей в нейронных сетях. Правило распространения сигналов в сети.
2. Примеры применения перцептронов.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра ИУС

Утверждено на заседании кафедры  
« \_\_\_\_ » \_\_\_\_\_ 200\_г.  
Заведующий кафедрой  
Утверждаю \_\_\_\_\_

Факультет МиИ  
Курс II  
Дисциплина  
Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 9

1. Структура связей. Правило комбинирования входящих сигналов.
2. Самоорганизующаяся карта признаков. Базовая архитектура сети.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры  
« \_\_\_\_ » \_\_\_\_\_ 200\_г.  
Заведующий кафедрой  
Утверждаю \_\_\_\_\_

Кафедра ИУС  
Факультет МиИ  
Курс II  
Дисциплина  
Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 10

1. Свойства биологических нейронных сетей.
2. Многослойный перцептрон. Базовая архитектура сети.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры  
« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Кафедра ИУС  
Факультет МиИ  
Курс II

Заведующий кафедрой  
Утверждаю \_\_\_\_\_

Дисциплина  
Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 11

1. Свойства искусственных нейронных сетей.
2. Алгоритм обратного распространения ошибки.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры  
« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой  
Утверждаю \_\_\_\_\_

Кафедра ИУС  
Факультет МиИ  
Курс II  
Дисциплина  
Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 12

1. Недостатки и ограничения нейронных сетей.
2. Сети с радиальными и базисными функциями.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра ИУС

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой

Утверждаю \_\_\_\_\_

Факультет МиИ

Курс II

Дисциплина

Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 13

1. Этапы решения задачи с помощью нейронной сети.
2. Самоорганизующаяся карта признаков. Свойства самоорганизующейся карты признаков.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_г.

Заведующий кафедрой

Утверждаю \_\_\_\_\_

Кафедра ИУС

Факультет МиИ

Курс II

Дисциплина

Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 14

1. Формальный нейрон. Виды функций активации.
2. Сеть Хопфилда. Вычислительная энергия сети Хопфилда.

### АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры

« \_\_\_\_ » \_\_\_\_\_ 200\_ г.

Заведующий кафедрой

Утверждаю \_\_\_\_\_

Кафедра ИУС

Факультет МиИ

Курс II

Дисциплина

Нейроинформатика

### ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 15

1. Проблемы, возникающие при моделировании нейронных сетей.
2. Обучение по алгоритму обратного распространения ошибки.

### 13. КАРТА КАДРОВОЙ ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ

Лектор – доцент Акилова Ирина Михайловна.

Руководитель практических работ – доцент Акилова Ирина Михайловна.

Руководитель лабораторных работ – доцент Акилова Ирина Михайловна.

### СОДЕРЖАНИЕ

1. Рабочие программы	3
3. График самостоятельной работы студентов	23
4. Методические рекомендации по проведению самостоятельной работы студентов	24
5. Перечень учебников, учебных пособий	25
6. Краткий конспект лекций	26
7. Методические указания по выполнению лабораторных работ	87
8. Методические указания по выполнению практических работ	109
9. Методические указания по организации	

межсессионного контроля знаний студентов	152
10. Фонд контрольных заданий для оценки качества знаний по дисциплине	153
11. Контрольные вопросы к зачету	158
10. Комплект экзаменационных билетов	159
13. Карта кадровой обеспеченности дисциплины	167

Ирина Михайловна Акилова,  
*доцент кафедры ИиУС АмГУ*

Учебно-методический комплекс по дисциплине «Нейроинформатика»

---

Изд-во АмГУ. Подписано к печати  
Тираж                      Заказ

Формат 60x84/16. Усл. печ. л .