

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Магнитогорский государственный технический университет им. Г.И. Носова»



## РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

### МЕТОДЫ НЕЙРОКОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

Направление подготовки  
09.03.01 Информатика и вычислительная техника

Профиль программы  
Программное обеспечение средств вычислительной техники и автоматизированных систем

Уровень высшего образования – бакалавриат  
Программа подготовки – академический бакалавриат

Форма обучения  
заочная

Институт  
Кафедра  
Курс

*энергетики и автоматизированных систем  
вычислительной техники и программирования*  
5

Магнитогорск  
2017 г.

Рабочая программа составлена на основе ФГОС ВО по направлению подготовки (специальности) 09.03.01 Информатика и вычислительная техника, утвержденного приказом МО и Н РФ от 12.01.2016 № 5.

Рабочая программа рассмотрена и одобрена на заседании кафедры вычислительной техники и программирования «26» сентября 2017 г., протокол № 2.

Зав. кафедрой  / О.С. Логунова/

Рабочая программа одобрена методической комиссией института энергетики и автоматизированных систем «27» сентября 2017 г., протокол № 2.

Председатель  / С.И. Лукьянов/

Рабочая программа составлена:

доцентом каф. ВТиП

 М.В.Зарецким

Рецензент:


начальник отдела инновационных разработок ЗАО «КонсОмСКС», канд. техн. наук

 / А.Н. Панов/

## Лист актуализации рабочей программы


---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2017-2018 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 26 09 2017г. № 2  
Зав. кафедрой  О.С. Логунова

---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2018 - 2019 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 5 09 2018г. № 1  
Зав. кафедрой  О.С. Логунова

---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2019 - 2020 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 19 09 2019г. № 5  
Зав. кафедрой  О.С. Логунова

---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2020 - 2021 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 19 09 2020г. № 5  
Зав. кафедрой  О.С. Логунова

### 1 Цели освоения дисциплины (модуля)

Целями освоения дисциплины (модуля) «Методы нейροкомпьютерного моделирования» являются:

- формирование у студентов понимания основных парадигм нейροинформатики;
- выработка у студентов умения применять нейросетевые методы для решения практических задач;
- выработка понимания сложностей, связанных с реализацией нейросетевых методологий и путей их преодоления;
- выработка навыков применения современных программных средств, реализующих нейросетевые методы.

Для достижения поставленных целей в курсе «Методы нейροкомпьютерного моделирования» решаются задачи:

- изучение методологических основ нейροкомпьютерного моделирования;
- изучение математических основ нейροкомпьютерного моделирования;
- освоение современного программного обеспечения, реализующего методы нейροкомпьютерного моделирования;

### 2 Место дисциплины (модуля) в структуре образовательной программы подготовки бакалавра (магистра, специалиста)

Дисциплина «Методы нейροкомпьютерного моделирования» входит в вариативную часть блока 1 образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения следующих дисциплин:

- философии (базовая часть блока 1 образовательной программы). Знания, полученные при изучении данной дисциплины, позволят обучающимся освоить основы эпистемологии, необходимые для понимания нейροкомпьютерной парадигмы в моделировании;
- математики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, позволят обучающимся освоить математический аппарат нейροкомпьютерного моделирования;
- информатики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для освоения средств обработки информации в соответствии с нейροкомпьютерной парадигмой;
- прикладного программирования (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для освоения методологии разработки программ в нейροкомпьютерной парадигме.

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для выполнения выпускной квалификационной работы.

### 3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Методы нейροкомпьютерного моделирования» обучающийся должен обладать следующими компетенциями:

Структурный элемент компетенции	Планируемые результаты обучения
ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.	
Знать	– основные парадигмы моделирования - детерминированная модель, веро-

Структурный элемент компетенции	Планируемые результаты обучения
	ятностная модель, нейросетевая модель; – методы построения моделей в условиях неустранимой неопределенности; – методы построения нейросетевых моделей, устойчивых к естественным и искусственным помехам.
Уметь	– определять целесообразность применения нейросетевой методологии для моделирования явления или процесса; – выбирать наиболее подходящие для создания модели нейросетевые архитектуры; – модифицировать архитектуру искусственной нейронной сети в соответствии с требованиями адекватности модели.
Владеть	– навыками применения нейросетевых средств моделирования.
Структурный элемент компетенции	Планируемые результаты обучения
ПК-3 Обладает способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности	
Знать	– основы методологии построения нейросетевых баз знаний, систем поддержки принятия решений для создания моделей предметной области; – методологию верификации результатов моделирования, осуществляемого с использованием нейросетевых интеллектуальных систем; – методологию разработки систем поддержки принятия решений.
Уметь	– выбирать концепцию построения модели интеллектуальной системы поддержки принятия решений, соответствующую поставленной прикладной задаче; – выбирать алгоритмы верификации функционирования моделей на основе нейросетевых интеллектуальных систем.
Владеть	– навыками применения программного обеспечения интеллектуальных систем для разработки интеллектуальных моделей; – навыками осуществления настройки и верификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей; – навыками осуществления модификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей.

#### 4 Структура и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 4 зачетные единицы 144 акад. часа, в том числе:

- контактная работа – 8,7 акад. часа:
  - аудиторная – 8 акад. часов;
  - внеаудиторная – 0,7 акад. часа
- самостоятельная работа – 131,4 акад. часа;
- контроль – 3,9 акад. часа;

Раздел/ тема дисциплины	Курс	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
1. Раздел. Основные парадигмы нейрокомпьютерного моделирования.	5							
1.1. Тема. Нейронные сети в природе и технике.	5	0,5	0,5		10	Самостоятельное изучение учебной и научной литературы.	Беседа – обсуждение. Устный опрос.	<i>ПК-2 –зув</i> <i>ПК-3 –зув</i>
1.2. Тема. Обзор нейросетевых архитектур, применяемых в моделировании.	5	0,5	0,5		10	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	<i>ПК-2 –зув</i> <i>ПК-3 –зув</i>
<b>Итого по разделу</b>	<b>5</b>	<b>1</b>	<b>1</b>		<b>20</b>		<b>Проверка контрольной работы</b>	
2. Раздел. Перцептронные модели	5							
2.1. Тема. Простейшие модели на основе перцептрона Ф. Розенблатта. Проблема	5	0,5	0,5		15	Самостоятельное изучение учебной и научной литературы.	Беседа – обсуждение. Анализ программного кода.	<i>ПК-2– зув</i> <i>ПК-3– зув</i>

Раздел/ тема дисциплины	Курс	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
XOR.						Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Устный опрос.	
2.2. Тема. Модели на основе многослойного персептрона	5	0,5	0,5		15	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	<i>ПК-2 – зув</i> <i>ПК-3 – зув</i>
<b>Итого по разделу</b>	<b>3</b>	<b>1</b>	<b>1</b>		<b>30</b>		<b>Проверка контрольной работы</b>	
3. Раздел. Ассоциативные модели.	5					Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	<i>ПК-2 – зув</i> <i>ПК-3 – зув</i>
3.1. Тема. Модели на основе автоассоциативных сетей.	5	0,5	0,5/0,5И		20	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	<i>ПК-2 – зув</i> <i>ПК-3 – зув</i>

Раздел/ тема дисциплины	Курс	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
3.2. Тема. Модели на основе гетероассоциативных сетей.	5	0,5	0,5/0,5И		20	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	<i>ПК-2 –зув</i> <i>ПК-3 –зув</i>
<b>Итого по разделу</b>	<b>5</b>	<b>1</b>	<b>1/ИИ</b>		<b>40</b>		<b>Проверка контрольной работы</b>	
4. Раздел. Радиально-базисные модели.	5							
4.1. Тема. Модели с нулевой ошибкой.	5	0,5	0,5/0,5И		20	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	<i>ПК-2 –зув</i> <i>ПК-3 –зув</i>
4.2. Тема. Записи, классы и объекты в языке LISP. Синтез функциональной и объектно-ориентированной парадигм программирования. Диалект VisualLISP для системы компьютерной графики AutoCAD.	5	0,5	0,5/0,5И		21,4	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	<i>ПК-2 –зув</i> <i>ПК-3 –зув</i>
<b>Итого по разделу</b>	<b>5</b>	<b>1</b>	<b>1/ИИ</b>		<b>41,4</b>		<b>Проверка контрольной работы</b>	



Раздел/ тема дисциплины	Курс	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
<b>Итого по курсу</b>	<b>5</b>	<b>4</b>	<b>4/2И</b>		<b>131,4</b>		<b>Зачет</b>	
<b>Итого по дисциплине</b>	<b>3</b>	<b>4</b>	<b>4/2И</b>		<b>131,4</b>			

## 5 Образовательные и информационные технологии

1. **Традиционные образовательные технологии** ориентируются на организацию образовательного процесса, предполагающую прямую трансляцию знаний от преподавателя к студенту (преимущественно на основе объяснительно-иллюстративных методов обучения). Учебная деятельность студента носит в таких условиях, как правило, репродуктивный характер.

### **Формы учебных занятий с использованием традиционных технологий:**

Информационная лекция – последовательное изложение материала в дисциплинарной логике, осуществляемое преимущественно вербальными средствами (монолог преподавателя).

Семинар – беседа преподавателя и студентов, обсуждение заранее подготовленных сообщений по каждому вопросу плана занятия с единым для всех перечнем рекомендуемой обязательной и дополнительной литературы.

Практическое занятие, посвященное освоению конкретных умений и навыков по предложенному алгоритму.

Лабораторная работа – организация учебной работы с реальными материальными и информационными объектами, экспериментальная работа с аналоговыми моделями реальных объектов.

2. **Технологии проблемного обучения** – организация образовательного процесса, которая предполагает постановку проблемных вопросов, создание учебных проблемных ситуаций для стимулирования активной познавательной деятельности студентов.

3. **Интерактивные технологии** – организация образовательного процесса, которая предполагает активное и нелинейное взаимодействие всех участников, достижение на этой основе лично значимого для них образовательного результата. Наряду со специализированными технологиями такого рода принцип интерактивности прослеживается в большинстве современных образовательных технологий. Интерактивность подразумевает субъект - субъектные отношения в ходе образовательного процесса и, как следствие, формирование саморазвивающейся информационно-ресурсной среды.

### **Формы учебных занятий с использованием специализированных интерактивных технологий:**

Лекция «обратной связи» – лекция–провокация (изложение материала с заранее запланированными ошибками), лекция-беседа, лекция-дискуссия, лекция–пресс-конференция.

4. **Информационно-коммуникационные образовательные технологии** – организация образовательного процесса, основанная на применении специализированных программных сред и технических средств работы с информацией.

## 6 Учебно-методическое обеспечение самостоятельной работы обучающихся

### **Задание к лабораторной работе по теме:**

#### **Нейронные сети в природе и в технике.**

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием.

```
import numpy as np
```

```
def nonlin(x, deriv=False):  
    if deriv:  
        return nonlin(x)*(1-nonlin(x))  
    return 1/(1+np.exp(-x))  
  
def train(X,y,n_iter):  
    np.random.seed(1)  
    syn0=2*np.random.random((3,1))-1
```

```

for iter in range(n_iter):
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))
    l1_error=y-l1
    l1_delta=l1_error*nonlin(l1,True)
    syn0 += np.dot(l0.T,l1_delta)
return syn0
def query(syn,XX):
    res = nonlin(np.dot(XX,syn))
    return res

```

Задания.

1.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]].T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans);

```

2.

```

X = np.array([[1,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]].T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 12000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans);

```

3.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]].T
XX = np.array([[0,0,3],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 15000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

4.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]].T
XX = np.array([[0,1,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 17000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

5.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1])).T
XX = np.array([[0,0,2],[0,2,2],[2,1,2],[2,2,2]])
n_iter = 9000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

6.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1])).T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[2,1,7]])
n_iter = 20000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

7.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1])).T
XX = np.array([[0,0,2],[0,2,2],[2,5,2],[1,2,2]])
n_iter = 11000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

8.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1])).T
XX = np.array([[0,0,2],[0,2,2],[2,5,2],[2,2,2]])
n_iter = 19000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

9.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1])).T
XX = np.array([[1,1,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

10.

```

X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1])).T
XX = np.array([[1,1,2],[0,2,2],[2,0,2],[5,2,2]])

```

```

n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

**Задание к лабораторной работе по теме:**

**Обзор нейросетевых архитектур, применяемых в моделировании**

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

Проанализировать результаты.

1.

```

function q=my_Prog_01()
P = [1 -1.2]
T=[0.5 1]
net=newlind(P,T)
Y=sim(net,P)
net.IW{1,1}
q=net.b

```

2.

```

function q=my_Prog_02()
Diap_Entry_Changes = [-1 1;-1 1];
Number_of_Neurons =1;
Initial_Weights = [2 3];
Initial_Bias=[-4];
P=[5;6];
net=newlin(Diap_Entry_Changes,Number_of_Neurons);
net.IW{1,1}=Initial_Weights;
net.b{1}=Initial_Bias;
q=sim(net,P)

```

3.

```

function q=my_Prog_03()
P=[1 -1.2];
T=[0.5 1];
w_range=-1:0.1:0; b_range=0.5:0.1:1;
ES=errsurf(P,T,w_range,b_range,'purelin')
contour(w_range,b_range,ES,20)
hold on
plot(-2.2727e-001,7.2727e-001,'x')
hold off

```

4.

```

function q5=my_Prog_04
P=[1 -1.2];
T=[0.5 1];
maxlr=0.4*maxlinlr(P,'bias')
w_range=-1:0.1:0; b_range=0.5:0.1:1;
ES=errsurf(P,T,w_range,b_range,'purelin')
contour(w_range,b_range,ES,20)

```

```
hold on
plot(-2.2727e-001,7.2727e-001,'x')
hold off
```

5.

```
function q=my_Prog_05()
Diap_Entry_Changes = [-1 -1;-1 1];
Number_of_Neurons = 1;
Initial_Weights = [3 3];
Initial_Bias=[-4];
P=[5;6];
net=newlin(Diap_Entry_Changes,Number_of_Neurons);
net.IW{1,1}=Initial_Weights;
net.b{1}=Initial_Bias;
a=sim(net,P)
```

6.

```
function q=my_Prog_06()
Diap_Entry_Changes = [1 1]
Number_of_Neurons = 1
Input_Delay_Vector = [1 1];
Initial_Weights = [1 2];
Initial_Bias_Connect=0
P={-1 -1/2 1/2 1};
Q=[-1 -1/3 1/4 1];
R={[-1 1] [-1/2 1/2] [1/2 -1/2] [1 -1]};
net=newlin(Diap_Entry_Changes,Number_of_Neurons,Input_Delay_Vector)
net.IW{1,1}=Initial_Weights
net.biasConnect = Initial_Bias_Connect;
a=sim(net,P)
b=sim(net,Q)
c=sim(net,R)
q=cell2mat(c(1,:))
```

7.

```
function a=my_Prog_07()
Diap_Entry_Changes = [-1 0;-1 0];
Number_of_Neurons = 1;
Initial_Weights = [2 3];
Initial_Bias=[-4];
P=[5;6];
net=newlin(Diap_Entry_Changes,Number_of_Neurons);
net.IW{1,1}=Initial_Weights;
net.b{1}=Initial_Bias;
a=sim(net,P)
```

8.

```
function q=my_Prog_08()
Diap_Entry_Changes = [-1 1]
Number_of_Neurons = 1
Input_Delay_Vector = [0 1];
Initial_Weights = [1 2];
Initial_Bias_Connect=0
```

```

P={-1 -1/2 1/2 1};
Q=[-1 -1/2 1/2 1];
R=[[-1 1] [-1/2 1/2] [1/2 -1/3] [1 -1]];
net=newlin(Diap_Entry_Changes,Number_of_Neurons,Input_Delay_Vector)
net.IW{1,1}=Initial_Weights
net.biasConnect = Initial_Bias_Connect;
q=sim(net,P)
b=sim(net,Q)
c=sim(net,R)
q=cell2mat(c(1,:))
end

```

9.

```

function q=my_Prog_09()
P=[1 -1.2];
T=[0.5 1];
w_range=-1:0.01:0; b_range=0.5:0.01:1;
ES=errsurf(P,T,w_range,b_range,'purelin')
contour(w_range,b_range,ES,20)
hold on
plot(-2.2727e-001,7.2727e-001,'x')
hold off

```

10.

```

function q=my_Prog_10()
P=[3 -1.-2];
T=[0.5 3];
maxlr=0.4*maxlinlr(P,'bias')
w_range=-1:0.01:0; b_range=0.5:0.1:1;
ES=errsurf(P,T,w_range,b_range,'purelin')
contour(w_range,b_range,ES,20)
hold on
plot(-2.2727e-001,7.2727e-001,'x')
hold off

```

Задание к лабораторной работе по теме:

### **Простейшие модели на основе перцептрона Ф. Розенблатта. Проблема XOR**

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием. Путь к файлам с исходными данными пользователь задает самостоятельно.

```

import numpy as np
from numpy.random import seed
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

class Perceptron(object):
    def __init__(self, eta=0.01, n_iter=10):
        self.eta=eta
        self.n_iter=n_iter

    def fit(self, X,y):

```

```

self.w_ = np.zeros(1 +X.shape[1])
self.errors_ = []

for _ in range(self.n_iter):
    errors = 0
    for Xi, target in zip(X,y):
        update = self.eta * (target-self.predict(Xi))
        self.w_[1:] += update*Xi
        self.w_[0] += update
        errors += int(update != 0.0)
    self.errors_.append(errors)
return self
def net_input(self, X):
    return np.dot(X,self.w_[1:]) + self.w_[0]

def predict(self, X):
    return np.where(self.net_input(X) >=0,1,-1)

```

1.

```

fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.1,n_iter=10)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()

```

2.

```

fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.05,n_iter=50)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()

```



3.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.09,n_iter=170)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

4.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.2,n_iter=7)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

5.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.09,n_iter=15)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

```

6.
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='magenta', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.08,n_iter=10)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()

7.
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.05,n_iter=100)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()

8.
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.06,n_iter=1000)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()

```

9.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.07,n_iter=110)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

10.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.03,n_iter=190)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

### **Задание к лабораторной работе по теме:**

#### **Модели на основе многослойного персептрона**

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

Проанализировать результаты.

1.

```
function MLP0
x=0:pi/10:2*pi;
y=(sin(2*x +pi/4)+1).*wxp(-x.^2);
S1=5;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
```

```

PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=3000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

2.

```

function MLP0
x=0:0.01:4;
y=(s(2*pi*x)+0.5).*wxp(-x.^2);
S1=7;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 150;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=2000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

3.

```

function MLP0
x=0:0.01:4;
y=(sin(2*pi*x)+1).*wxp(-x.^2);
S1=4;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=2000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

4.

```

function MLP0
x=0:0.01:4;
y=(sin(2*pi*x)+0.1).*wxp(-x.^3);
S1=5;
S2=1;
PR=minmax(x);

```

```

TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=2000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

5.

```

function MLP0
x=0:0.01:5;
y=(sin(2*pi*x)+1).*wexp(-x.^2);
S1=6;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=2000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

6.

```

function MLP0
x=0:0.01:4;
y=(sin(2*pi*x)+1).*wexp(-x.^2);
S1=4;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.05;
net.adaptParam.epochs=3000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

7.

```

function MLP0
x=0:0.01:45

```

```

y=(sin(2*pi*x)+1).*wxp(-x.^2);
S1=5;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=2000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

8.

```

function MLP0
x=0:0.05:6;
y=(sin(2*pi*x)+1).*wxp(-x.^2);
S1=7;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=10000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

9.

```

function MLP0
x=0:0.05:4;
y=(sin(2*pi*x)+1).*wxp(-x.^2);
S1=5;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],{TF1,TF2},BTF, BLF, PF);
net.adaptParam.show = 100;
net.adaptParam.Ir = 0.2;
net.adaptParam.epochs=2000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

```

10.
function MLP_0
x=0:0.01:4;
y=(cos(2*pi*x)+1).*wexp(-x.^2);
S1=5;
S2=1;
PR=minmax(x);
TF1='radbas';
TF2='purelin';
BTF='trainlm';
BLF='learngd';
PF='mse';
net = newff(PR,[S1 S2],[TF1,TF2],BTF, BLF, PF);
net.adaptParam.show = 50;
net.adaptParam.Ir = 0.1;
net.adaptParam.epochs=3000;
[net stats] = adapt(net,p,t)
sim(net,p)

```

### **Задание к лабораторной работе по теме:**

#### **Модели на основе автоассоциативных сетей**

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

Проанализировать результаты.

```

1.
function hopf()
T = [-1 -1 1;1 -1 1]';
net = newhop(T);
Ai = T;
[Y Pf Af] = sim(net,2,[],Ai);
Y
Ai = {[-0.9;-0.8;0.7]};
[Y Pf Af] = sim(net,{1 5},{},Ai);
Y{1}
Ai = {[1.1;-0.7;0.75]};
[Y Pf Af] = sim(net,{1 10},{},Ai);
Y{1}

```

```

2.
function hopf()
% Создаем сеть Хопфилда с 2 точками равновесия в 3-мерном пространстве
T = [1 -1;-1 1;1 1; -1 -1]';
plot(T(1,:),T(2,:),'*r')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);

```

```

Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

3.

```

function hopf()
T = [-1 -1;-1 1;1 1; -1 -1]';
plot(T(1,:),T(2,:),'*r')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

4.

```

function hopf()
T = [1 -1;-1 1;1 1; -1 1]';
plot(T(1,:),T(2,:),'*r')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);

```



```

% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

5.

```

function hopf()
% Создаем сеть Хопфилда с 2 точками равновесия в 3-мерном пространстве
T = [1 -1;1 1;1 1; -1 -1]';
plot(T(1,:),T(2:,:),'*r')
axis([-1.1 -1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
% Моделируем работу сети
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

6.

```

function hopf()
T = [1 -1;-1 -1;1 1; -1 -1]';
plot(T(1,:),T(2:,:),'*r')
axis([-1.1 1.1 1.1 -1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

7.

```

function hopf()
% Создаем сеть Хопфилда с 2 точками равновесия в 3-мерном пространстве
T = [1 -1;-1 1;-1 1; -1 -1]';
plot(T(1,:),T(2:,:),'*r')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];

```

```

    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

8.

```

function hopf()
T = [-1 -1;-1 1;1 1; -1 -1]';
plot(T(1,:),T(2:,:),'*r')
axis([-1.1 -1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

9.

```

function hopf()
T = [-1 -1;-1 1;1 -1; -1 -1]';
plot(T(1,:),T(2:,:),'*r')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);

```

```

[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

10.

```

function hopf()
T = [1 -1;-1 1;1 1; -1 -1]';
plot(T(1,:),T(2:,:),'*r')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai);
for i = 1:25
    a = {rands(2,1)};
    [Y Pf Af] = sim(net,{1 20},{},a);
    record = [cell2mat(a), cell2mat(Y)];
    start = cell2mat(a);
    plot(start(1,1), start(2,1),'kx',record(1,:), record(2,:))
end

```

### **Задание к лабораторной работе по теме:**

Модели на основе гетероассоциативных сетей.

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием.

```
class HammingNeuron:
```

```

    def __init__(self, weights, next_neuron=None):
        self.weights = list()
        self.inputs = list()
        self.next_neuron = None
        for w in weights:
            self.weights.append(w)
            self.inputs.append(0)
        self.next_neuron = next_neuron

```

```

def change_weight(self, ind_of_weight, new_value):
    self.weights[ind_of_weight] = new_value

def set_input(self, ind_of_input, value):
    self.inputs[ind_of_input] = value

def set_next_neuron(self, next_neuron):
    self.next_neuron = next_neuron

def count_output(self):
    res = 1/2 + sum(self.inputs[i] * self.weights[i] for i in range(0, len(self.weights)))/(2 *
len(self.weights))
    return res

def get_output(self):
    self.next_neuron.set_value(self.count_output())

```

class MaxNetNeuron:

```

def __init__(self, index, weights, next_neuron):
    self.value = 0
    self.reinitial_value = 0
    self.inputs = list()
    self.weights = list()
    self.layer_neurons = list()
    self.index = index
    self.next_neuron = next_neuron
    for w in weights:
        self.weights.append(w)
        self.inputs.append(None)

def set_layer_neurons(self, layer_neurons):
    for n in layer_neurons:
        self.layer_neurons.append(n)

def set_value(self, value):
    self.value = value
    self.reinitial_value = value

def set_only_current_value(self, value):
    self.value = value

def set_input(self, ind_of_neuron, value):
    self.inputs[ind_of_neuron] = value

def count_output(self):
    # if first time
    if self.inputs[self.index] is None:
        return self.value
    # if not first time
    else:
        return self.inputs[self.index] - \

```

```
sum(self.inputs[i] * self.weights[i] for i in range(0, len(self.weights)) if i !=
self.index)
```

```
def recount_value(self):
    self.value = self.count_output()

def get_output_inside_layer(self):
    for n in self.layer_neurons:
        n.set_input(self.index, self.value)

def get_output(self):
    self.next_neuron.set_value(self.value)

def reinitialize_neuron(self):
    self.value = self.reinitial_value
    for i in range(0, len(self.inputs)):
        self.inputs[i] = None
```

```
class ThresholdNeuron:
```

```
def __init__(self, index, next_neurons):
    self.value = None
    self.next_neurons = list()
    self.index = index
    for n in next_neurons:
        self.next_neurons.append(n)

def set_value(self, value):
    self.value = value

def count_output(self):
    if self.value > 0:
        return 1
    else:
        return 0

def get_output(self):
    for n in self.next_neurons:
        n.set_input(self.index, self.count_output())
```

```
class OutputNeuron:
```

```
def __init__(self, weights):
    self.weights = list()
    self.inputs = list()
    for w in weights:
        self.weights.append(w)
        self.inputs.append(0)

def set_input(self, index, value):
    self.inputs[index] = value
```

```
def get_output(self):
    return sum(self.weights[i] * self.inputs[i] for i in range(0, len(self.weights)))
```

class HammingLayer:

```
def __init__(self, weights, next_neurons):
    self.neurons = list()
    for i in range(0, len(weights)):
        new_neuron = HammingNeuron(weights[i], next_neurons[i])
        self.neurons.append(new_neuron)
```

```
def run(self, inputs):
    for n in self.neurons:
        for i in range(0, len(inputs)):
            n.set_input(i, inputs[i])
    for n in self.neurons:
        n.get_output()
```

class MaxNetLayer:

```
def __init__(self, next_neurons):
    self.neurons = list()
    k = len(next_neurons)
    for i in range(0, k):
        weights = [random.random() * 1/(k - 1) for j in range(0, i)] + [1] + \
            [random.random() * 1/(k - 1) for j in range(i + 1, k)]
        new_neuron = MaxNetNeuron(i, weights, next_neurons[i])
        self.neurons.append(new_neuron)
    for n in self.neurons:
        n.set_layer_neurons(self.neurons)
```

```
def run(self):
    for n in self.neurons:
        n.get_output_inside_layer()
    for n in self.neurons:
        n.recount_value()
    for n in self.neurons:
        n.get_output()
```

```
def reinitialize_layer(self, num_of_not_null_neuron, eps):
    self.neurons[num_of_not_null_neuron].reinitial_value -= eps
    for n in self.neurons:
        n.reinitialize_neuron()
```

class ThresholdLayer:

```
def __init__(self, count, next_neurons):
    self.neurons = list()
    for i in range(0, count):
```

```

        new_neuron = ThresholdNeuron(i, next_neurons)
        self.neurons.append(new_neuron)

def run(self):
    for n in self.neurons:
        n.get_output()

def get_first_not_null_element(self):
    for n in self.neurons:
        if n.count_output() == 1:
            return self.neurons.index(n)

class OutputLayer:

def __init__(self, weights):
    self.neurons = list()
    for i in range(0, len(weights[0])):
        self.neurons.append(OutputNeuron([weights[j][i] for j in range(0, len(weights))]))

def get_result(self):
    l = []
    for n in self.neurons:
        l.append(n.get_output())
    return l

class HammingNetwork:
    """
    Initial arguments:
    learning_examples - list of learning examples
    eps - maximal distance between winners
    max_count_of_outputs - maximal count of winners
    """

def __init__(self, learning_examples, eps, max_count_of_outputs):
    self.max_count_of_outputs = max_count_of_outputs
    self.eps = eps
    self.output_layer = OutputLayer(learning_examples)
    self.threshold_layer = ThresholdLayer(len(learning_examples),
self.output_layer.neurons)
    self.max_net_layer = MaxNetLayer(self.threshold_layer.neurons)
    self.hamming_layer = HammingLayer(learning_examples,
self.max_net_layer.neurons)

def classification(self, example_inputs):
    res = []
    self.hamming_layer.run(example_inputs)
    first_time = True
    while(True):
        while(True):
            self.max_net_layer.run()
            if sum(n.count_output() for n in self.threshold_layer.neurons) == 1:

```



```

        break
    for n in self.max_net_layer.neurons:
        if n.next_neuron.count_output() == 0:
            n.set_only_current_value(0)
    self.threshold_layer.run()
    res.append(self.output_layer.get_result())
    if first_time:
        first_time = False

self.max_net_layer.reinitialize_layer(self.threshold_layer.get_first_not_null_element(), self.eps)
    continue
else:
    if res[len(res) - 1] in res[0:len(res) - 1] or len(res) > self.max_count_of_outputs:
        res = res[0:len(res) - 1]
        return res
    else:

self.max_net_layer.reinitialize_layer(self.threshold_layer.get_first_not_null_element(), self.eps)
    continue

```

```

1.
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, -1, 1, 1,
-1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                       '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                       '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('-----')

```

```

2.

if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                       '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                       '9': [1, 1, 1, 1, -1, -1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.4
    max_count_of_output = 3

```

```

my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
for ex in my_validation_examples:
    print('example')
    print(str(ex))
    for ans in my_hamming_network.classification(dict_of_numbers[ex]):
        print('answer')
        print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
    print('-----')

```

3.

```

if __name__ == '__main__':
dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                    '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                    '6': [1, 1, -1, 1, 1, 1, 1], '7': [1, 1, 1, -1, 1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                    '9': [1, 1, 1, 1, -1, 1, 1]}
my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
my_validation_examples = ['6', '7', '8']
my_eps = 0.3
max_count_of_output = 3
my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
for ex in my_validation_examples:
    print('example')
    print(str(ex))
    for ans in my_hamming_network.classification(dict_of_numbers[ex]):
        print('answer')
        print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
    print('-----')

```

4.

```

if __name__ == '__main__':
dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                    '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                    '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                    '9': [1, 1, 1, 1, -1, 1, 1]}
my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
my_validation_examples = ['6', '7', '8']
my_eps = 0.3
max_count_of_output = 3
my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
for ex in my_validation_examples:
    print('example')
    print(str(ex))
    for ans in my_hamming_network.classification(dict_of_numbers[ex]):
        print('answer')
        print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
    print('-----')

```

5.

```

if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                        '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                        '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                        '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('-----')

```

6.

```

if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, 1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
1, 1],
                        '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                        '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                        '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('-----')

```

7.

```

if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                        '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                        '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                        '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in

```

```

my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('-----')

8.
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1,1,1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1, -
1, 1],
                        '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                        '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                        '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('-----')

9.
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                        '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                        '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                        '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('-----')

10.
if __name__ == '__main__':

```

```

dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                  '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                  '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                  '9': [1, 1, 1, 1, -1, 1, 1]}
my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
my_validation_examples = ['6', '7', '8']
my_eps = 0.3
max_count_of_output = 3
my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
for ex in my_validation_examples:
    print('example')
    print(str(ex))
    for ans in my_hamming_network.classification(dict_of_numbers[ex]):
        print('answer')
        print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
    print('-----')

```

### **Задание к лабораторной работе по теме:**

#### **Модели с нулевой ошибкой**

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

```

1.
function RB()
P = -5:0.1:5;
T = (P.^2+1)./(P.^2+4)+3*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off

```

```

2.
function RB()
P = -5:0.1:5;
T = cos(3*P/2)+sin(2*P)+3*randn;
SPREAD = 0.7;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off

```

3.

```
function RB()
P = -5:0.1:5;
T = (P.^2+7)./(P.^4+2)+3*randn;
SPREAD = 0.4;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off
```

4.

```
function RB()
P = -7:0.1:7;
T = (P.^6+5)./(P.^2+4)+3*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -9:0.01:9;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off
```

5.

```
function RB()
P = -5:0.1:5;
T = sin(P/4)+cos(2*P)+3*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off
```

6.

```
function RB()
P = -5:0.1:5;
T = (P.^2+1)./(P.^4+7)+7*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
```

```

plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off

```

7.

```

function RB()
P = -5:0.1:5;
T = (P.^2+1)./(P.^2+4)+3*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off

```

8.

```

function RB()
P = -5:0.1:5;
T = sin(3*P/4)+cos(1,5*P)+5*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off

```

9.

```

function RB()
P = -7:0.05:5;
T = (P.^4+5)./(P.^2+4)+3*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off

```

```

10.
function RB()
P = -5:0.1:5;
T = cos(3*P)+sin(P/2)+2*randn;
SPREAD = 0.5;
net = newrbe(P,T,SPREAD);
figure(1),clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on
X = -7:0.01:7;
Y = sim(net,X);
plot(X,Y,'LineWidth',2)
grid on
hold off

```

**Задание к лабораторной работе по теме:  
 Модели обобщенной регрессии**

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

```

1.
function GR()
P=1:8;
T=[0 1 2 3 2 1 2 1];
spread=0.7;
net = newgrnn(P,T,spread);
net.layers{1}.size
V = sim(net, P);
disp(V-T)

```

```

2.
function GR()
P=1:7;
T=[0 1 2 3 2 1 1];
spread=0.9;
net.layers{1}.size
V = sim(net, P);
V-T

```

```

3.
function GR()
P=1:8;
T=[0 1 2 3 2 1 2 1];
spread=0.7;
net = newgrnn(P,T,spread);
net.layers{1}.size
V = sim(net, P);
V-T

```

```

4.
function GR()
P=1:8;
T=[0 1 2 3 2 1 2 0];

```



```
spread=0.8;
net = newgrnn(P,T,spread);
net.layers{1}.size
V = sim(net, P);
V-T
```

```
5.
function GR()
P=1:9;
T=[0 1 2 4 3 1 2 1 1];
spread=0.7;
net = newgrnn(P,T,spread);
net.layers{1}.size
V = sim(net, P);
V-T
```

```
6.
function GR()
P=1:9;
T=[0 1 2 3 2 1 2 1 1];
spread=0.85;
net = newgrnn(P,T,spread);
net.layers{1}.size
V = sim(net, P);
V-T
```

```
7.
function GR()
P=1:10;
T=[0 1 2 3 2 1 2 2 1 1];
spread=0.7;
net.layers{1}.size
V = sim(net, P);
V-T
```

```
8.
function GR()
P=1:7;
T=[0 1 2 3 2 1 1];
spread=0.7;
net = newgrnn(P,T,spread);
net.layers{1}.size
V = sim(net, P);
V-T
```

```
9.
function GR()
P=1:8;
T=[0 1 2 3 2 1 2 1];
spread=0.7;
net = newgrnn(P,T,spread);
net.layers{1}.size
V = sim(net, P);
```

V-T

```
10.  
function GR()  
P=1:9;  
T=[0 1 1 2 3 2 1 2 1];  
spread=0.7;  
net = newgrnn(P,T,spread);  
net.layers{1}.size  
V = sim(net, P);  
V-T
```

### Контрольная работа № 1.

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием.

```
import numpy as np  
  
def nonlin(x, deriv=False):  
    if (deriv):  
        return nonlin(x)*(1-nonlin(x))  
    return 1/(1+np.exp(-x))  
  
def train(X,y,n_iter):  
    np.random.seed(1)  
    syn0=2*np.random.random((3,1))-1  
    for iter in range(n_iter):  
        l0 = X  
        l1 = nonlin(np.dot(l0,syn0))  
        l1_error=y-l1  
        l1_delta=l1_error*nonlin(l1,True)  
        syn0 += np.dot(l0.T,l1_delta)  
    return syn0  
def query(syn,XX):  
    res = nonlin(np.dot(XX,syn))  
    return res  
  
1.  
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])  
y = np.array([[0,1,1,1]].T  
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[2,2,2]])  
n_iter = 1000  
syn=train(X,y,n_iter)  
print(syn)  
ans=query(syn,XX)  
print(ans)  
  
2.  
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])  
y = np.array([[0,0,1,1]].T  
XX = np.array([[0,1 ,2],[0,2,2],[2,0,2],[2,2,2]])  
n_iter = 15000  
syn=train(X,y,n_iter)
```

```
print(syn)
ans=query(syn,XX)
print(ans)
```

```
3.
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,4],[0,3,2],[2,0,2],[2,3,2]])
n_iter = 11000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

```
4.
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,2,1],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

```
5.
X = np.array([[1,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,3,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

```
6.
X = np.array([[0,0,1],[1,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,3,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

```
7.
X = np.array([[0,0,1],[0,1,1],[1,0,1],[0,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[3,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

```

8.
X = np.array([[1,0,1],[0,1,1],[1,0,0],[1,0,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

```

9.
X = np.array([[0,1,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,3,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

```

10.
X = np.array([[0,1,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,1,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

```

### **Контрольная работа № 2.**

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

```

1.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
net.trainParam.passes=100
p={ [3;2] [1;-2] [-2;2] [-1;1] }
t={ 0 1 0 1 }
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

2.
function Perc()
PR =[-2 2;-2 2]

```

```

S=1
net = newp(PR, S)
net.trainParam.passes=170
p={ [2;2] [1;-2] [-2;2] [-2;1] }
t={ 0 1 0 1 }
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

3.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
net.trainParam.passes=100
p={ [2;-2] [1;-2] [-2;2] [-1;1] }
t={ 0 1 1 1 }
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

4.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
% Создаем сеть
net.trainParam.passes=100
p={ [2;2] [1;-2] [-2;2] [-1;1] }
t={ 0 1 0 1 }
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

5.
function Perc()
PR =[-2 2;-2 2]
S=1 % Число нейронов в слое

```

```

net = newp(PR, S)
net.trainParam.passes=220
p={[2;2] [1;-2] [-1;2] [-1;1]}
t={0 1 0 1}
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

6.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
net.trainParam.passes=250
p={[-1;2] [1;-2] [-2;2] [-1;1]}
t={0 1 0 1}
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

7.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
net.trainParam.passes=270
p={[2;2] [1;-2] [-2;2] [-2;1]}
t={0 1 0 1}
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

8.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
net.trainParam.passes=190

```

```

p={ [2;1] [1;-2] [-2;2] [-1;1] }
t={ 0 1 0 1 }
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

9.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
net.trainParam.passes=210
p={ [2;2] [1;-2] [-2;1] [-1;1] }
t={ 0 1 0 1 }
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

```

10.
function Perc()
PR =[-2 2;-2 2]
S=1
net = newp(PR, S)
net.trainParam.passes=260
p={ [2;2] [1;-2] [-2;1] [-1;1] }
t={ 0 1 0 1 }
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)
[net a e] = adapt(net,p,t)
twts=net.IW{1,1}
tbias=net.b{1}
a1=sim(net,p)

```

### Контрольная работа №3.

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

```

1.
function hopf()
T = [1 -1;-1 1;1 1; -1 -1]';
net = newhop(T);
W = net.LW{1,1}

```

```

b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)

```

```

2.
function hopf()
T = [1 -1;-1 1;1 1; 1 -1]';
net = newhop(T);
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
% Моделируем работу сети
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)

```

```

3.
function hopf()
T = [1 1;-1 1;1 1; 1 -1]';
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
% Моделируем работу сети
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)

```

```

4.
function hopf()
T = [-1 -1;-1 1;1 -1; -1 -1]';
net = newhop(T);

```



```

W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)

```

```

5.
function hopf()
T = [1 -1;-1 1;1 -1; -1 -1]';
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
% Моделируем работу сети
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)

```

```

6.
function hopf()
% Создаем сеть Хопфилда с 2 точками равновесия в 3-мерном пространстве
T = [1 -1;-1 1;1 1; -1 -1]';
plot(T(1,:),T(2:,:),'*r')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield')
xlabel('a(1)'), ylabel('a(2)')
% Считаем веса и смещения сети Хопфилда
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2:,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')

```

```
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)
```

7.

```
function hopf()
T = [1 -1;1 1;1 1; -1 1]';
net = newhop(T);
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
% Моделируем работу сети
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)
```

8.

```
function hopf()
T = [-1 -1;-1 1;1 1; -1 -1]';
net = newhop(T);
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
% Моделируем работу сети
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)
```

9.

```
function hopf_2
T = [1 -1;-1 -1;1 -1; 1 -1]';
net = newhop(T);
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
% Моделируем работу сети
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
```

```
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)
```

10.

```
function hopf()
T = [1 -1;-1 -1;1 -1; -1 1]';
net = newhop(T);
% Создали сеть
W = net.LW{1,1}
b = net.b{1,1}
Ai = T;
[Y Pf Af] = sim(net,4,[],Ai);
Y
Pf
Af
plot(T(1,:),T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
net = newhop(T);
[Y Pf Af] = sim(net,4,[],Ai)
```

#### **Контрольная работа № 4.**

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

1.

```
function RB()
P=-1:0.1:1;
T=[-0.8702 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
    0.3449 0.1916 -0.0312 -0.2189 -0.3201];
GOAL = 0.01;
SPREAD = 1;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end
```

2.

```
function RB()
P=-1:0.1:1;
T=[-0.5533 -0.5470 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
    0.3449 0.1816 -0.0312 -0.3127 -0.3201];
GOAL = 0.07;
SPREAD = 9.9
net = newrb(P,T,GOAL, SPREAD);
```

```

net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

3.
function RB()
P=-1:0.1:1;
T=[-0.9602 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
    0.3449 0.1816 -0.0223 -0.2189 0.44101];
GOAL = 0.07;
SPREAD = 3;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

4.
function RB()
P=-1:0.1:1;
T=[-0.9602 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
    0.3449 0.1816 -0.0312 -0.2189 -0.3201];
GOAL = 0.01;
SPREAD = 1;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

5.
function RB()
P=-1:0.1:1;
T=[-0.3365 0.5770 -0.0729 0.3771 0.3305 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
    0.3449 0.1816 -0.0312 -0.2189 -0.3201];
GOAL = 0.03;
SPREAD = 3;

```

```

net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

6.
function RB()
P=-1:0.1:1;
T=[-0.7755 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
    0.3449 0.1816 -0.0312 -0.2189 -0.3201];
GOAL = 0.009;
SPREAD = 2;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

7.
function RB()
P=-1:0.1:1;
T=[-0.9602 -0.4532 -0.0729 0.3771 -0.8709 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
    0.3449 0.1816 -0.0312 -0.2189 -0.3201];
GOAL = 0.07;
SPREAD = 3;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

8.
function RB()
P=-1:0.1:1;
T=[0.5532 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
    -0.2013 -0.4344 -0.5000 -0.3930 -0.8947 0.0988 0.3072 0.3960 ...
    0.3449 0.1816 -0.0312 -0.2189 -0.3201];
GOAL = 0.08;

```

```

SPREAD = 2;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

9.
function R_B_3
P=-1:0.1:1;
T=[0.2219 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
   -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
   0.3449 0.1816 -0.3129 -0.2189 -0.3201];
GOAL = 0.0089;
SPREAD = 3;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

```

10.
function R_B_3
P=-1:0.1:1;
T=[-0.9602 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609 0.1336 ...
   -0.2013 -0.4344 -0.5000 -0.3930 -0.1647 0.0988 0.3072 0.3960 ...
   0.3449 0.1816 -0.0312 -0.4478 0.2215];
GOAL = 0.07;
SPREAD = 3;
net = newrb(P,T,GOAL, SPREAD);
net.layers{1}.size
plot(P,T,'+k','MarkerSize',4,'LineWidth',2)
hold on
X = -1:0.1:1;
Y = sim(net, X);
plot(X,Y,'ob','MarkerSize',5,'LineWidth',2)
hold off
end

```

## 7 Оценочные средства для проведения промежуточной аттестации

### а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства																																								
<b>ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.</b>																																										
Знать	<ul style="list-style-type: none"> <li>– основные парадигмы моделирования</li> <li>- детерминированная модель, вероятностная модель, нейросетевая модель;</li> <li>– методы построения моделей в условиях неустранимой неопределенности;</li> <li>– методы построения нейросетевых моделей, устойчивых к естественным и искусственным помехам.</li> </ul>	<p>Список теоретических вопросов:</p> <ul style="list-style-type: none"> <li>– моделирование с помощью простейшего построение перцептрона;</li> <li>– реакция однослойного перцептрона на предъявление задачи XOR;</li> <li>– моделирование на основе многослойных перцептронов;</li> <li>– решение задачи XOR для многослойных перцептронов;</li> <li>– определение зависимости качества обучения многослойного перцептрона от его топологии</li> </ul>																																								
Уметь	<ul style="list-style-type: none"> <li>– определять целесообразность применения нейросетевой методологии для моделирования явления или процесса;</li> <li>– выбирать наиболее подходящие для создания модели нейросетевые архитектуры;</li> <li>– модифицировать архитектуру искусственной нейронной сети в соответствии с требованиями адекватности модели.</li> </ul>	<p>Список практических заданий:</p> <ul style="list-style-type: none"> <li>– для автоматического обнаружения садовых вредителей двух типов используется нейрокомпьютерная система распознавания, основанная на экспертных знаниях. Вредители имеют два определяющих параметра, измеряемые в миллиметрах: расстояние от головы до кончика хвоста («длину») и расстояние от кончика правого крыла до кончика левого крыла («ширину»). На основании мнений экспертов сформирована выборка:</li> </ul> <table border="1" data-bbox="976 1114 1525 1270"> <tbody> <tr> <td>Особь</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> </tr> <tr> <td>Длина</td> <td>10</td> <td>7</td> <td>5</td> <td>5</td> <td>8</td> <td>4</td> <td>4</td> </tr> <tr> <td>Ширина</td> <td>5</td> <td>3</td> <td>11</td> <td>3</td> <td>4</td> <td>2</td> <td>4</td> </tr> <tr> <td>Тип</td> <td>1</td> <td>1</td> <td>2</td> <td>1</td> <td>2</td> <td>1</td> <td>2</td> </tr> </tbody> </table> <p>Выбрать архитектуру нейросети, провести обучение. С помощью обученной нейросети отнести к одному из классов особь, имеющую «длину» 8 и «ширину» 7 миллиметров.</p> <ul style="list-style-type: none"> <li>– в условиях той же задачи провести обучение по выборке:</li> </ul> <table border="1" data-bbox="976 1422 1525 1458"> <tbody> <tr> <td>Особь</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> </tr> </tbody> </table>	Особь	1	2	3	4	5	6	7	Длина	10	7	5	5	8	4	4	Ширина	5	3	11	3	4	2	4	Тип	1	1	2	1	2	1	2	Особь	1	2	3	4	5	6	7
Особь	1	2	3	4	5	6	7																																			
Длина	10	7	5	5	8	4	4																																			
Ширина	5	3	11	3	4	2	4																																			
Тип	1	1	2	1	2	1	2																																			
Особь	1	2	3	4	5	6	7																																			

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства																								
		<table border="1" data-bbox="976 312 1525 427"> <tr> <td>Длина</td> <td>10</td> <td>2</td> <td>7</td> <td>8</td> <td>8</td> <td>4</td> <td>4</td> </tr> <tr> <td>Ширина</td> <td>6</td> <td>4</td> <td>3</td> <td>6</td> <td>4</td> <td>2</td> <td>7</td> </tr> <tr> <td>Тип</td> <td>1</td> <td>1</td> <td>2</td> <td>1</td> <td>2</td> <td>1</td> <td>2</td> </tr> </table> <p data-bbox="976 432 2168 612">Выбрать архитектуру нейросети, провести обучение. С помощью обученной нейросети отнести к одному из классов особь, имеющую «длину» 8 и «ширину» 7 миллиметров. Объяснить существенное различие в архитектуре нейросетей, пригодных для решения предложенных задач.</p>	Длина	10	2	7	8	8	4	4	Ширина	6	4	3	6	4	2	7	Тип	1	1	2	1	2	1	2
Длина	10	2	7	8	8	4	4																			
Ширина	6	4	3	6	4	2	7																			
Тип	1	1	2	1	2	1	2																			
Владеть	– навыками применения нейросетевых средств моделирования.	<p data-bbox="976 624 2168 651">Список комплексных заданий:</p> <ul data-bbox="976 660 2168 1027" style="list-style-type: none"> <li>– пусть количество вредителей в саду равно <math>N</math>. Для уничтожения одного вредителя типа 1 требуется затратить <math>S</math> руб., для уничтожения одного вредителя типа 2 требуется затратить <math>T</math> руб. Сгенерировать список вредителей («длину» и «ширину» каждого из них считать случайной величиной, равномерно распределенной в диапазоне от 1 до 10). С помощью обеих нейросетей, описанных в предыдущем разделе, определить тип каждого из вредителей, оценить стоимость их уничтожения. Принять: <math>N=700</math>, <math>S=0,005</math> руб., <math>T=0,007</math> руб.</li> <li>– в условиях предыдущей задачи сгенерировать <math>M</math> выборок длиной <math>N</math>. По результатам анализа всех выборок составить прогнозную модель, предсказывающую стоимость борьбы с вредителями. Принять <math>M=100</math>.</li> </ul>																								
<b>ПК-3 Обладает способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности</b>																										
Знать	<ul data-bbox="387 1118 954 1447" style="list-style-type: none"> <li>– основы методологии построения нейросетевых баз знаний, систем поддержки принятия решений для создания моделей предметной области;</li> <li>– методологию верификации результатов моделирования, осуществляемого с использованием нейросетевых интеллектуальных систем;</li> <li>– методологию разработки систем под-</li> </ul>	<p data-bbox="976 1118 2168 1145">Список теоретических вопросов:</p> <ul data-bbox="976 1155 2168 1385" style="list-style-type: none"> <li>– моделирование на основе сети Хемминга;</li> <li>– моделирование на основе сети Хопфилда;</li> <li>– моделирование на основе RBF-сети;</li> <li>– моделирование на основе GRNN-сети;</li> <li>– моделирование на основе сети Кохонена;</li> <li>– работа с системами поддержки принятия решений.</li> </ul>																								



Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства																																								
	держки принятия решений.																																									
Уметь	<ul style="list-style-type: none"> <li>– выбирать концепцию построения модели интеллектуальной системы поддержки принятия решений, соответствующую поставленной прикладной задаче;</li> <li>– выбирать алгоритмы верификации функционирования моделей на основе нейросетевых интеллектуальных систем.</li> </ul>	<p>Список практических заданий:</p> <ul style="list-style-type: none"> <li>– добавка витаминной смеси в корм для цыплят увеличивает средний суточный привес. В приведены данные о количестве добавки (граммы на килограмм кормов) и добавке к среднему привесу (граммы).</li> </ul> <table border="1" data-bbox="976 523 1509 603"> <tr> <td>Добавка</td> <td>3</td> <td>7</td> <td>12</td> <td>18</td> <td>25</td> </tr> <tr> <td>Привес</td> <td>70</td> <td>110</td> <td>130</td> <td>140</td> <td>155</td> </tr> </table> <p>Составить несколько нейросетевых прогнозных моделей с использованием перцептронов и нейросетей обобщенной регрессии.</p> <ul style="list-style-type: none"> <li>– торговая фирма оценивает эффективность затрат на рекламу с помощью ботов. Товары делятся на следующие ценовые категории: премиальные, высокой стоимости, эконом. Данные даются следующими таблицами:</li> </ul> <table border="1" data-bbox="976 788 2159 1166"> <thead> <tr> <th>Товар</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th></th> </tr> </thead> <tbody> <tr> <td>Тип</td> <td>Прем.</td> <td>Эконом</td> <td>Высокой ст.</td> <td>Эконом</td> <td>Прем.</td> <td>Высокой ст.</td> </tr> <tr> <td>Затраты на рекламу (руб)</td> <td>150000</td> <td>90000</td> <td>120000</td> <td>70000</td> <td>200000</td> <td>110000</td> </tr> <tr> <td>Увеличение объема продаж (руб)</td> <td>320000</td> <td>4700000</td> <td>2800000</td> <td>5700000</td> <td>900000</td> <td>2500000</td> </tr> </tbody> </table> <p>Расклассифицировать товары по эффективности рекламы с помощью ботов. В одну группу могут попасть товары из различных ценовых групп.</p>	Добавка	3	7	12	18	25	Привес	70	110	130	140	155	Товар	1	2	3	4	5		Тип	Прем.	Эконом	Высокой ст.	Эконом	Прем.	Высокой ст.	Затраты на рекламу (руб)	150000	90000	120000	70000	200000	110000	Увеличение объема продаж (руб)	320000	4700000	2800000	5700000	900000	2500000
Добавка	3	7	12	18	25																																					
Привес	70	110	130	140	155																																					
Товар	1	2	3	4	5																																					
Тип	Прем.	Эконом	Высокой ст.	Эконом	Прем.	Высокой ст.																																				
Затраты на рекламу (руб)	150000	90000	120000	70000	200000	110000																																				
Увеличение объема продаж (руб)	320000	4700000	2800000	5700000	900000	2500000																																				
Владеть	<ul style="list-style-type: none"> <li>– навыками применения программного обеспечения интеллектуальных систем для разработки интеллектуальных моделей;</li> <li>– навыками осуществления настройки</li> </ul>	<p>Список комплексных заданий:</p> <ul style="list-style-type: none"> <li>– В условиях задачи 1 из предыдущего пункта установить наиболее приемлемое с экономической точки зрения количество витаминной смеси. Считаем, что один грамм смеси стоит 0, 05руб., прибавка 1 г. живого веса приносит 0,1 руб. прибыли. Суточную потребность в кормах считаем равной 5000 кг.</li> </ul>																																								

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства																																		
	<p>и верификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей;</p> <p>– навыками осуществления модификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей.</p>	<p>– в условиях второй задачи рассмотреть дополнительно эффективность рекламы в социальных сетях. Сведения об эффективности рекламы в социальных сетях даны в следующей таблице:</p> <table border="1" data-bbox="974 422 2159 799"> <thead> <tr> <th data-bbox="974 422 1167 459">Товар</th> <th data-bbox="1167 422 1328 459">1</th> <th data-bbox="1328 422 1496 459">2</th> <th data-bbox="1496 422 1664 459">3</th> <th data-bbox="1664 422 1832 459">4</th> <th data-bbox="1832 422 1995 459">5</th> <th data-bbox="1995 422 2159 459"></th> </tr> <tr> <th data-bbox="974 459 1167 536">Тип</th> <td data-bbox="1167 459 1328 536">Прем.</td> <td data-bbox="1328 459 1496 536">Эконом</td> <td data-bbox="1496 459 1664 536">Высокой ст.</td> <td data-bbox="1664 459 1832 536">Эконом</td> <td data-bbox="1832 459 1995 536">Прем.</td> <td data-bbox="1995 459 2159 536">Высокой ст.</td> </tr> <tr> <th data-bbox="974 536 1167 647">Затраты на рекламу (руб)</th> <td data-bbox="1167 536 1328 647">15000</td> <td data-bbox="1328 536 1496 647">10000</td> <td data-bbox="1496 536 1664 647">12000</td> <td data-bbox="1664 536 1832 647">15000</td> <td data-bbox="1832 536 1995 647">20000</td> <td data-bbox="1995 536 2159 647">1100</td> </tr> <tr> <th data-bbox="974 647 1167 799">Увеличение объема продаж (руб)</th> <td data-bbox="1167 647 1328 799">22000</td> <td data-bbox="1328 647 1496 799">1200000</td> <td data-bbox="1496 647 1664 799">230000</td> <td data-bbox="1664 647 1832 799">150000</td> <td data-bbox="1832 647 1995 799">600000</td> <td data-bbox="1995 647 2159 799">2500</td> </tr> </thead></table> <p>Расклассифицировать товары по совместной эффективности в обеих группах рекламы.</p>							Товар	1	2	3	4	5		Тип	Прем.	Эконом	Высокой ст.	Эконом	Прем.	Высокой ст.	Затраты на рекламу (руб)	15000	10000	12000	15000	20000	1100	Увеличение объема продаж (руб)	22000	1200000	230000	150000	600000	2500
Товар	1	2	3	4	5																															
Тип	Прем.	Эконом	Высокой ст.	Эконом	Прем.	Высокой ст.																														
Затраты на рекламу (руб)	15000	10000	12000	15000	20000	1100																														
Увеличение объема продаж (руб)	22000	1200000	230000	150000	600000	2500																														

## **б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:**

Промежуточная аттестация по дисциплине «Методы нейрокомпьютерного моделирования» основана на проверке выполнения практических заданий, в ходе которой выявляется степень сформированности умений и владений. Аттестация проводится в форме зачета.

### **Показатели и критерии оценивания зачета:**

– «зачтено» – обучающийся демонстрирует сформированность компетенций, умение применять изученный материал в практически важных ситуациях.

– «не зачтено» – обучающийся не может показать знания на уровне воспроизведения и объяснения информации, не может показать интеллектуальные навыки решения основных задач.

## **8 Учебно-методическое и информационное обеспечение дисциплины (модуля)**

### **а) Основная литература:**

1. Павлов, С.И. Системы искусственного интеллекта : учебное пособие / С.И. Павлов. – Томск : Томский государственный университет систем управления и радиоэлектроники, 2011. – Ч. 1. – 175 с. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=208933> (дата обращения: 30.10.2020). – ISBN 978-5-4332-0013-5. – Текст : электронный.

### **б) Дополнительная литература:**

1. Интеллектуальные информационные системы и технологии : учебное пособие / Ю.Ю. Громов, О.Г. Иванова, В.В. Алексеев и др. ; Тамбовский государственный технический университет. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2013. – 244 с. : ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=277713> (дата обращения: 30.10.2020). – Библиогр. в кн. – ISBN 978-5-8265-1178-7. – Текст : электронный.

### **в) Методические указания**

1. Жидьцов В.В. Практикум по нейросетевым технологиям. [Электронный ресурс]. Режим доступа: <http://bek.sibadi.org/fulltext/epd6.pdf>

### **г) Программное обеспечение и Интернет-ресурсы:**

*Программное обеспечение:* лицензионное программное обеспечение: операционная система; офисные программы; математические пакет, статистические пакеты, установленные на каждом персональном компьютере вычислительного центра ФГБОУ ВПО «МГТУ».

Перечень лицензионного программного обеспечения по ссылке:

<http://sps.vuz.magtu.ru/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2FShared%20Documents%2F%D0%9F%D0%BE%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0%20%D0%BA%20%D0%B0%D0%BA%D0%BA%D1%80%D0%B5%D0%B4%D0%B8%D1%82%D0%B0%D1%86%D0%B8%D0%B8%202020%2F%D0%A1%D0%B0%D0%BC%D0%BE%D0%BE%D0%B1%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%202019%D0%B3%2F%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5%20%D0%9F%D0%9E&InitialTabId=Ribbon.Document&VisibilityContext=WSSTabPersistence>

Официальные сайты промышленных предприятий и организаций: <http://www.mmk.ru>, <http://www.creditural.ru>, <http://www.magtu.ru>, <http://www.gks.ru> и т.п.; разработчиков программных продуктов: <http://www.statsoft.ru>, <http://www.microsoft.com>, <http://www.ptc.com> и т.п; сайты лабораторий компьютерной графики <http://graphics.cs.msu.ru> , <http://cgm.graphicon.ru>.

## **9 Материально-техническое обеспечение дисциплины (модуля)**

Материально-техническое обеспечение дисциплины включает:

Тип и название аудитории	Оснащение аудитории
Лекционная аудитория	Мультимедийные средства хранения, передачи и представления информации
Компьютерный класс	Персональные компьютеры с пакетом Office, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета
Аудитории для самостоятельной работы: компьютерные классы; читальные залы библиотеки	Все классы УИТ и АСУ с персональными компьютерами, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета
Аудиторий для групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации	Ауд. 282 и классы УИТ и АСУ
Помещения для самостоятельной работы обучающихся, оснащенных компьютерной техникой с возможностью подключения к сети «Интернет» и наличием доступа в электронную информационно-образовательную среду организации	Классы УИТ и АСУ
Помещения для хранения и профилактического обслуживания учебного оборудования	Центр информационных технологий – ауд. 379