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

УТВЕРЖДАЮ:
Директор института
_____ С.Н. Лукьянов
«27» сентября 2017 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

МЕТОДЫ НЕЙРОКОМПЬЮТРНОГО 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
Очная

Институт	*энергетики и автоматизированных систем*
Кафедра	*вычислительной техники и программирования*
Курс	*4*
Семестр	*7*

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/
(подпись) *(И.О. Фамилия)*

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 2017 г. № 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 2018 г. № 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 02 2020 г. № 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 02 2020 г. № 5
Зав. кафедрой _____ О.С. Логунова

1 Цели освоения дисциплины (модуля)

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

- формирование у студентов понимания основных парадигм нейроинформатики;
- 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 нейрокомпьютерного моделирования» решаются задачи:

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

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

Дисциплина «Методы нейрокомпьютерного 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 нейрокомпьютерной парадигмы в моделировании;
- математики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, позволят обучающимся освоить математический аппарат нейрокомпьютерного 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 нейрокомпьютерной 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 нейрокомпьютерной 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 нейрокомпьютерного 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.
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 – 55 акад. часов:
  - аудиторная – 54 акад. часа;
  - внеаудиторная – 1 акад. час
- самостоятельная работа – 89 акад. часов.

Раздел/ тема дисциплины	Семестр	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 1. Основные парадигмы нейрокомпьютерного моделирования.	7							
1.1. Тема. Основные парадигмы нейрокомпьютерного моделирования.	7	2	4		8	Самостоятельное изучение учебной и научной литературы.	Беседа – обсуждение. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*
1.2. Тема. Обзор нейросетевых архитектур, применяемых в моделировании	7	2	4		8	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*
Итого по разделу	**7**	**4**	**8**		**16**		**Проверка индивидуальных заданий**	
2. Раздел 2. Персептронные модели.	7							
2.1. Тема. Простейшие модели на основе персептрона Ф. Розенблатта. Проблема XOR	7	2	4		10	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*

Раздел/ тема дисциплины	Семестр	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				
						Выполнение лабораторной работы.		
2.2. Тема Модели на основе многослойного персептрона	7	2	4		10	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*
Итого по разделу	**7**	**4**	**8**		**20**		**Проверка индивидуальных заданий**	
3. Раздел. Ассоциативные модели.	7							
3.1. Тема. Модели на основе автоассоциативных сетей	7	2	4		12	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*
3.2. Тема. Модели на основе гетероассоциативных сетей.	7	2	4/2И		14	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*
Итого по разделу	**7**	**4**	**8/2И**		**26**		**Проверка индивидуальных заданий**	

Раздел/ тема дисциплины	Семестр	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				
4. Раздел. Радиально-базисные модели.	7							
4.1. Тема. Модели с нулевой ошибкой.	7	2	6/6И		12	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*
4.2. Тема. Модели обобщенной регрессии	7	4	6/6И		15	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув* *ПК3 – зув*
Итого по разделу		**6**	**12/12И**		**27**		**Проверка индивидуальных заданий**	
Итого за семестр		**18**	**36/14И**		**89**		**Зачет**	
Итого по дисциплине		**18**	**36/14И**		**89**			

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 личностно 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:

Основные парадигмы нейрокомпьютерного моделирования.

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_Entay_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 персептрона 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).*wxp(-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).*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 = 0.05;
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).*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)
```

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
```

Индивидуальные задания к разделу 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)
```

Индивидуальные задания к разделу 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)
```

Индивидуальные задания к разделу 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)
```

Индивидуальные задания к разделу 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:

<table>
<tr><th>Структурный элемент компетенции</th><th>Планируемые результаты обучения</th><th>Оценочные средства</th></tr>
<tr><td colspan="3">ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.</td></tr>
<tr><td>Знать</td><td>– основные парадигмы моделирования - детерминированная модель, вероятностная модель, нейросетевая модель;
– методы построения моделей в условиях неустранимой неопределенности;
– методы построения нейросетевых моделей, устойчивых к естественным и искусственным помехам.</td><td>Список теоретических вопросов:
– моделирование с помощью простейшего построение персептрона;
– реакция однослойного персептрона на предъявление задачи XOR;
– моделирование на основе ногослойных персептронов;
– решение задачи XOR для многослойных персептронов;
– определение зависимости качества обучения многослойного персептрона от его топологии</td></tr>
<tr><td>Уметь</td><td>– 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.</td><td>Список практических заданий:
– 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:
<table>
<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>
</table>
Выбрать архитектуру нейросети, провести обучение. С помощью обученной нейросети отнести к одному из классов особь, имеющую «длину» 8 и «ширину» 7 миллиметров.</td></tr>
</table>

Примечание [O1]:

Примечание [O2]:

Примечание [O3]: Должны быть конкретные задания

<table>
<tr><th>Структурный элемент компетенции</th><th>Планируемые результаты обучения</th><th>Оценочные средства</th></tr>
<tr><td></td><td></td><td>– в условиях той же задачи провести обучение по выборке:
<table>
<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>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>
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.</td></tr>
<tr><td>Владеть</td><td>– навыками применения нейросетевых средств моделирования.</td><td>Список комплексных заданий:
– пусть количество вредителей в саду равно N. Для уничтожения одного вредителя типа 1 требуется затратить S руб., для уничтожения одного вредителя типа 2 требуется затратить T 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: N=700, S=0,005 руб., T=0,007 руб.
– в условиях предыдущей задачи сгенерировать M выборок длиной N. По результатам анализа всех выборок составить прогнозную модель, предсказывающую стоимость борьбы с вредителями. Принять M=100.</td></tr>
<tr><td colspan="3">ПК-3 Обладает способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности</td></tr>
<tr><td>Знать</td><td>– основы методологии построения нейросетевых баз знаний, систем поддержки принятия решений для создания моделей предметной области;
– методологию верификации результатов моделирования, осуществляемого с использованием нейросетевых интеллек-</td><td>Список теоретических вопросов:
– моделирование на основе сети Хемминга;
– моделирование на основе сети Хопфилда;
– моделирование на основе RBF-сети;
– моделирование на основе GRNN-сети;
– моделирование на основе сети Кохонена;</td></tr>
</table>

<table>
<tr><td>Структурный элемент компетенции</td><td>Планируемые результаты обучения</td><td>Оценочные средства</td></tr>
<tr><td></td><td>туальных систем;
– методологию разработки систем поддержки принятия решений.</td><td>– работа с системами поддержки принятия решений.</td></tr>
<tr><td>Уметь</td><td>– выбирать концепцию построения модели интеллектуальной системы поддержки принятия решений, соответствующую поставленной прикладной задаче;
– выбирать алгоритмы верификации функционирования моделей на основе нейросетевых интеллектуальных систем.</td><td>Список практических заданий:
– 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).
<table>
<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>
Составить несколько нейросетевых прогнозных моделей с использованием персептронов и нейросетей обобщенной регрессии.
– торговая фирма оценивает эффективность затрат на рекламу с помощью ботов. Товары делятся на следующие ценовые категории: премиальные, высокой стоимости, эконом. Данные даются следующими таблицами:
<table>
<tr><td>Товар</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr>
<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>
</table>
Расклассифицировать товары по эффективности рекламы с помощью ботов. В одну группу могут попасть товары из различных ценовых групп.</td></tr>
<tr><td>Владеть</td><td>– навыками применения программного обеспечения интеллектуальных систем для разработки интеллектуальных моделей;</td><td>Список комплексных заданий:
– 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. Суточ-</td></tr>
</table>

<table>
<tr><th>Структурный элемент компетенции</th><th>Планируемые результаты обучения</th><th>Оценочные средства</th></tr>
<tr><td></td><td>– 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.</td><td>ную потребность в кормах считаем равной 5000 кг.
– в условиях второй задачи рассмотреть дополнительно эффективность рекламы в социальных сетях. Сведения об эффективности рекламы в социальных сетях даны в следующей таблице:
<table>
<tr><td>Товар</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr>
<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>15000</td><td>10000</td><td>12000</td><td>15000</td><td>20000</td><td>1100</td></tr>
<tr><td>Увеличение объема продаж (руб)</td><td>22000</td><td>1200000</td><td>230000</td><td>150000</td><td>600000</td><td>2500</td></tr>
</table>
Расклассифицировать товары по совместной эффективности в обеих группах рекламы.</td></tr>
</table>

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 У

~~8 У~~чебно-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, те-	Ауд. 282 и классы УИТ и АСУ

Тип и название аудитории	Оснащение аудитории
кущего контроля и промежуточной аттестации	
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