# **ЛАБОРАТОРНАЯ РАБОТА №2:** **«РЕШЕНИЕ ЗАДАЧИ РАСПОЗНАВАНИЯ С ПОМОЩЬЮ ПЕРСЕПТРОНА»**

**Цель:** Построить модель персептрона в системе MatLab 6.1 с помощью стандартных блоков (blockset) и ознакомиться со стандартной моделью персептрона из ППП Neural Network.

**Теоретическое введение:**

Простейшая из нейронных сетей - однослойный **персептрон**, веса и смещение которого могут быть настроены таким образом, чтобы решить задачу классификации входных векторов, что позволит в дальнейшем решать сложные проблемы анализа коммутационных соединений, распознавания образов и других задач классификации с высоким быстродействием и гарантией правильного результата.

По команде help percept можно получить следующую информацию об М-функциях, входящих в состав ППП Neural Network Toolbox и относящихся к построению нейронных сетей на основе персептронов:

Таблица 2.1

|  |  |
| --- | --- |
| **М-функция** | **Описание** |
| New networks | Формирование нейронной сети |
| newp | Создание персептрона |
| Using networks | Работа с нейронной сетью |
| sim | Моделирование сети |
| init | Инициализация сети |
| adapt | Адаптация сети |
| train | Обучение сети |
| Weight functions | Функции взвешивания |
| dotprod | Скалярное произведение |
| Net input functions | Функции накопления |
| Netsum | Сумма взвешенных входов |
| Transfer functions | Функции активации |
| Hardlim | Ступенчатая функция с жесткими ограничениями |
| hardlims | Симметричная ступенчатая функция с жесткими ограничениями |
| Initialization functions | Функции инициализации |
| Initlay | Инициализация слоев |
| initwb | Инициализация весов и смещений |
| Initzero | Инициализация нулевых весов и смещений |
| Performance functions | Функции оценки качества сети |
| mae | Средняя абсолютная погрешность |
| Learning functions | Функции настройки параметров персептрона |
|  | Окончание табл.2.1 |
| **М-функция** | **Описание** |
| Learning functions | Функции настройки параметров персептрона |
| Learnp | Абсолютная функция настройки |
| Learnpn | Нормированная функция настройки |
| Adapt functions | Функции адаптации |
| adaptwb | Адаптация весов и смещений |
| Training function | Функции обучения |
| Trainwb | Правило обучения весов и смещений |
| Demonstrations | Демонстрационные примеры |
| Demopl | Классификация с использованием персептрона с двумя входами |
| Demop2 | Классификация с использованием персептрона с тремя входами |
| Demop3 | Классификация с использованием персептрона двумя нейронами |
| Demop4 | Формирование входных векторов внешнего слоя |
| Demop5 | Обучение с использованием нормированной функции настройки |
| Demop6 | Пример линейно неразделимых векторов |
| Dernop7 | Классификация с использованием двухслойного персептрона |

**Нейрон персептрона**. Нейрон, используемый в модели персептрона, имеет ступенчатую функцию активации hardlim с жесткими ограничениями (рис. 2.1).

Каждый элемент вектора входа персептрона взвешен с соответствующим весом w1j, и их сумма является входом функции активации. Нейрон персептрона возвращает 1, если вход функции активации n≥0, и 0, если n< 0.



Рис. 2.1. Нейрон персептрона

Функция активации с жесткими ограничениями придает персептрону способность классифицировать векторы входа, разделяя пространство входов на 2 области, как это показано на рис. 2.2 для персептрона с двумя входами и смещением.



Рис. 2.2. Классификация точек на плоскости.

Пространство входов делится на 2 области разделяющей линией L, которая для двумерного случая задается уравнением

wTp + b = 0. (2.1)

Эта линия перпендикулярна к вектору весов **w** и смещена на величину b*.* Векторы входа выше линии L соответствуют положительному потенциалу нейрона, и, следовательно, выход персептрона для этих векторов будет равен 1; векторы входа ниже линии L соответствуют выходу персептрона, равному 0. При изменении значений смещения и весов граница линии L изменяет свое положение. Персептрон без смещения всегда формирует разделяющую линию, проходящую через начало координат; добавление смещения формирует линию, которая не проходит через начало координат, как это показано на рис. 2.2. В случае, когда размерность вектора входа превышает 2, разделяющей границей будет служить гиперплоскость.

Демонстрационная программа nnd4db наглядно иллюстрирует перемещение разделяющей линии при решении задачи классификации векторов входа.

**Архитектура сети**. Персептрон состоит из единственного слоя, включающего S нейронов, как это показано на рис. 2.3, *а* и *б в* виде соответственно развернутой и укрупненной структурных схем; веса wi,j - это коэффициенты передачи от j-говхода к i-му нейрону. Уравнение однослойного персептрона имеет вид:

a = f(Wp+b). (2.2)



Рис. 2.3. Развернутая (а) и укрупненная (б) структурная схема персептрона.

**Модель персептрона.** Для формирования модели однослойного персептрона предназначена функция newp

net = newp(PR, S)

со следующими входными аргументами: PR - массив минимальных и максимальных значений для R элементов входа размера Rx2; S - число нейронов в слое.

В качестве функции активации персептрона по умолчанию используется функция hardlim.

*Пример:*

Функция

net = newp([0 2],1);

создает персептрон с одноэлементным входом и одним нейроном; диапазон значений входа - [0 2].

Определим некоторые параметры персептрона, инициализируемые по умолчанию.

*Веса входов:*

inputweights = net.inputweights{1>1}

inputweights =

delays: 0

initFcn: 'initzero'

learn: 1

learnFcn: 'learnp'

learnParam: [ ]

size: [1 1]

userdata: [1x1 struct]

weightFcn: 'dotprod1

Заметим, что функция настройки персептрона по умолчанию learnp; вход функции активации вычисляется с помощью функции скалярного произведения dotprod; функция инициализации initzero используется для установки нулевых начальных весов.

*Смещения:*

biases = net.biases{1}

biases =

initFcn: 'initzero'

learn: 1

learnFcn: 'learnp'

learnParam: []

size: 1

userdata: [1x1 struct]

Нетрудно увидеть, что начальное смещение также установлено в 0.

**Моделирование персептрона**

Рассмотрим однослойный персептрон с одним двухэлементным вектором входа, значения элементов которого изменяются в диапазоне от -2 до 2:

net = newp([-2 2;-2 2],1); % Создание персептрона net

По умолчанию веса и смещение равны 0, и для того, чтобы установить желаемые значения, необходимо применить следующие операторы:

net.IW{1,1}= [-1 1];

net.b {1} = [1];

В этом случае разделяющая линия имеет вид:

L: -p1+p2+l =0.

Это соответствует рис. 2.1.

#### Структурная схема модели персептрона показана на рис. 2.4



Рис. 2.4. Структурная схема модели персептрона.

Теперь определим, как откликается сеть на входные векторы р1 и р2, расположенные по разные стороны от разделяющей линии:

p1 = [1; 1];

al = sim(net,pl) % Моделирование сети net с входным вектором р1

al = 1

р2 = [1; -1];

а2 = sim(net,p2) % Моделирование сети net с входным вектором р2

а2 = 0

Персептрон правильно классифицировал эти 2 вектора.

Заметим, что можно было бы ввести последовательность двух векторов в виде массива ячеек и получить результат также в виде массива ячеек:

% Последовательность двух векторов в виде массива ячеек

р3 = {[1; 1] [1; -1]};

а3 = sim(net,p3) % Моделирование сети net при входном сигнале р3

а3 = [1] [0]

**Инициализация параметров.** Для однослойного персептрона в качестве параметров нейронной сети в общем случае выступают веса входов и смещения. Допустим, что создается перcептрон с двухэлементным вектором входа и одним нейроном:

net = newp([-2 2;-2 2],1);

Запросим характеристики весов входа:

net. Inputweights {1, 1}

ans =

delays: 0

initFcn: 'initzero'

learn: 1

learnFcn: 'learnp'

learnParam: [ ]

size: [1 2]

userdata: [1xl struct]

weightFcn: ‘dotprod’

Из этого перечня следует, что в качестве функции инициализации по умолчанию используется функция initzero, которая присваивает весам входа нулевые значения. В этом можно убедиться, если извлечь значения элементов матрицы весов и смещения:

wts = net.IW{l,l}

wts =0 0

bias = net.b{l}

bias =0

Теперь переустановим значения элементов матрицы весов и смещения:

net.IW{1,1} = [3, 4]

net.b{1} = 5

wts =3 4

bias = 5

Для того, чтобы вернуться к первоначальным установкам параметров персептрона, и предназначена функция init:

net = init(net);

wts

wts =0 0

bias

bias = 0

Можно изменить способ, каким инициализируется персептрон с помощью функции init. Для этого достаточно изменить тип функций инициализации, которые применяются для установки первоначальных значений весов входов и смещений. Например, воспользуемся функцией инициализации rands, которая устанавливает случайные значения параметров персептрона.

% Задать функции инициализации весов и смещений

net.inputweights{1,1}.initFcn = 'rands';

net.biases{l}.initFcn = 'rands';

% Выполнить инициализацию ранее созданной сети с новыми функциями

net = init(net);

wts = net.IW{l,l}

wts = -0.96299 0.64281

bias = net.b{1}

bias = -0.087065

Видно, что веса и смещения выбраны случайным образом.

**Процедуры настройки параметров.** Определим процесс обучения персептрона как процедуру настройки весов и смещений с целью уменьшить разность между желаемым (целевым) и истинным сигналами на его выходе, используя некоторое ***правило настройки (обучения)****.* Процедуры обучения делятся на 2 класса: ***обучение с учителем***и ***обучение без учителя****.*

При обучении с учителем задается множество примеров требуемого поведения сети, которое называется ***обучающим множеством***

{p1,t1},{p2,t2 ,},...,{pQ,tQ}. (2.3)

Здесь p1, р2,..., рQ - входы персептрона, a t1, t2,..., tQ *-* требуемые (целевые) выходы.

При подаче входов выходы персептрона сравниваются с целями. Правило обучения используется для настройки весов и смещений персептрона так, чтобы приблизить значение выхода к целевому значению. Алгоритмы, использующие такие правила обучения, называются ***алгоритмами обучения с учителем****.* Для их успешной реализации необходимы эксперты, которые должны предварительно сформировать обучающие множества. Разработка таких алгоритмов рассматривается как первый шаг в создании систем искусственного интеллекта.

В этой связи ученые не прекращают спора на тему, можно ли считать алгоритмы обучения с учителем естественными и свойственными природе, или они созданы искусственны. Например, обучение человеческого мозга, на первый взгляд, происходит без учителя: на зрительные, слуховые, тактильные и прочие рецепторы поступает информация извне и внутри мозга происходит некая самоорганизация. Однако нельзя отрицать и того, что в жизни человека немало учителей - и в буквальном, и в переносном смысле, -которые координируют реакции на внешние воздействия. Вместе с тем, как бы ни развивался спор приверженцев этих двух концепций обучения, представляется, что обе они имеют право на существование. И рассматриваемое нами правило обучения персептрона относится к правилу обучения с учителем.

При обучении без учителя веса и смещения изменяются только в связи с изменениями входов сети. В этом случае целевые выходы в явном виде не задаются. Главная черта, делающая обучение без учителя привлекательным, - это его самоорганизация, обусловленная, как правило, использованием обратных связей. Что касается процесса настройки параметров сети, то он организуется с использованием одних и тех же процедур. Большинство алгоритмов обучения без учителя применяется при решении задач кластеризации данных, когда необходимо разделить входы на конечное число классов.

Что касается персептронов, то хотелось бы надеяться, что в результате обучения может быть построена такая сеть, которая обеспечит правильное решение, когда на вход будет подан сигнал, отличающийся от тех, которые использовались в процессе обучения.

**Правила настройки.** Настройка параметров (обучение) персептрона осуществляется с использованием обучающего множества. Обозначим через р вектор входов персептрона, а через t - вектор соответствующих желаемых выходов. Цель обучения – уменьшить погрешность е = а - t, которая равна разности между реакцией нейрона **а** и вектором цели **t.**

Правило настройки (обучения) персептрона должно зависеть от величины погрешности е. Вектор цели t может включать только значения 0 и 1, поскольку персептрон с функцией активации hardlim может генерировать только такие значения.

При настройке параметров персептрона без смещения и с единственным нейроном возможны только 3 ситуации:

1. Для данного вектора входа выход персептрона правильный (а=t и е = t - а= 0) и тогда вектор весов w не претерпевает изменений.
2. Выход персептрона равен 0, а должен быть равен 1 (а = 0, t=l и е=t = t-0=l). В этом случае вход функции активации wTp отрицательный и его необходимо скорректировать. Добавим к вектору весов w вектор входа р, и тогда произведение

(wT+ рT) р = wTp + рTр

изменится на положительную величину, а после нескольких таких шагов вход функции активации станет положительным и вектор входа будет классифицирован правильно. При этом изменятся настройки весов.

1. Выход нейрона равен 1, а должен быть равен 0 *(а* = 0, *t =* 1 и *е* = *t - а=* -1). В этом случае вход функции активации wTp положительный и его необходимо скорректировать. Вычтем из вектора весов w вектор входа р, и тогда произведение

(wT — рT ) р = wTp – рTр

изменится на отрицательную величину, а после нескольких шагов вход функции активации станет отрицательным, и вектор входа будет классифицирован правильно. При этом изменятся настройки весов.

Описанные соотношения положены в основу алгоритма настройки параметров персептрона, который реализован ППП Neural Network Toolbox в виде М-функции learnp. Каждый раз при выполнении функции learnp будет происходить перенастройка параметров персептрона. Доказано, что если решение существует, то процесс обучения персептрона сходится за конечное число итераций. Если смещение не используется, функция leamp ищет решение, изменяя только вектор весов w. Это приводит к нахождению разделяющей линии, перпендикулярной вектору w и которая должным образом разделяет векторы входа.

Рассмотрим простой пример персептрона с единственным нейроном и двухэлементным вектором входа:

net = newp([-2 2;-2 2],1);

Определим смещение b равным 0, а вектор весов w равным [1 -0.8]:

net.b{1} = 0;

W = [1 -0.8];

net.IW{l,l} = w;

Обучающее множество зададим следующим образом:

р = [1; 2];

t = [1];

Моделируя персептрон, рассчитаем выход и ошибку на первом шаге настройки (обучения):

а = sim(net.p)

а = 0

e = t-a

e = 1

Наконец, используя М-функцию настройки параметров learnp, найдем требуемое изменение весов:

dw = learnp(w,p, [ ],[ ],[ ],[ ],е,[ ],[ ], [ ] )

dw = 1 2

Тогда новый вектор весов примет вид:

w = w + dw

w = 2.0000 1.2000

Заметим, что описанные выше правило и алгоритм настройки (обучения) персептрона гарантируют сходимость за конечное число шагов для всех задач, которые могут быть решены с использованием персептрона. Это в первую очередь задачи классификации векторов, которые относятся к классу ***линейно отделимых****,* когда все пространство входов можно разделить на 2 области некоторой прямой линией, в многомерном случае - гиперплоскостью.

Демонстрационный пример nnd4pr позволяет выполнить многочисленные эксперименты по настройке (обучению) персептрона для решения задачи классификации входных векторов.

Для настройки (обучения) персептрона можно воспользоваться также М-функцией train. В этом случае используется все обучающее множество, и настройка параметров сети выполняется не после каждого прохода, а в результате всех проходов обучающего множества. К сожалению, не существует доказательства того, что такой алгоритм обучения персептрона является сходящимся. Поэтому использование М-функции train для обучения персептрона не рекомендуется.

Нейронные сети на основе персептрона имеют ряд ограничений. Во-первых, выход персептрона может принимать только одно из двух значений (0 или 1); во-вторых, персептроны могут решать задачи классификации только для линейно отделимых наборов векторов. Если с помощью прямой линии или гиперплоскости в многомерном случае можно разделить пространство входов на 2 области, в которых будут расположены векторы входа, относящиеся к различным классам, то векторы входа считаются линейно отделимыми. Если векторы входа линейно отделимы, то доказано, что при использовании процедуры адаптации задача классификации будет решена за конечное время. Если векторы входа линейно неотделимы, то процедура адаптации не в состоянии классифицировать все векторы должным образом. Демонстрационная программа demop6 иллюстрирует тщетность попытки классифицировать векторы входа, которые линейно неотделимы.

Для решения более сложных задач можно использовать сети с несколькими персептронами. Например, для классификации четырех векторов на 4 группы можно построить сеть с двумя персептронами, чтобы сформировать 2 разделяющие линии и таким образом приписать каждому вектору свою область.

Отметим еще одну особенность процесса обучения персептрона. Если длина некоторого вектора входа намного больше или меньше длины других векторов, то для обучения может потребоваться значительное время. Это обусловлено тем, что алгоритм настройки связан с добавлением или вычитанием входного вектора из текущего вектора весов. Таким образом, присутствие вектора входа с очень большими или малыми элементами может привести к тому, что потребуется длительное время для настройки параметров. Демонстрационная программа demop4 поясняет, как влияет выброс длины вектора на продолжительность обучения.

Можно сделать время обучения нечувствительным к большим или малым выбросам векторов входа, если несколько видоизменить исходное правило обучения персептрона:

 (2.4)

Действительно, из этого соотношения следует, что чем больше компоненты вектора входа р, тем большее воздействие он оказывает на изменение элементов вектора w. Можно уравновесить влияние больших или малых компонент, если ввести масштабирование вектора входа.

Решение состоит в том, чтобы нормировать входные данные так, чтобы воздействие любого вектора входа имело примерно равный вклад:

 (2.5)

Нормированное правило обучения персептрона реализуется М-функцией learnpn. Этот алгоритм требует несколько большего времени, но значительно сокращает количество циклов обучения, когда встречаются выбросы векторов входа. Демонстрационная программа demop5 иллюстрирует это правило обучения.

В заключение, следует отметить, что основное назначение персептронов - решать задачи классификации. Они великолепно справляются с задачей классификации линейно отделимых векторов; сходимость гарантируется за конечное число шагов. Длительность обучения чувствительна к выбросам длины отдельных векторов, но и в этом случае решение может быть построено. Однослойный персептрон может классифицировать только линейно отделимые векторы. Возможные способы преодолеть эту трудность предполагают либо предварительную обработку с целью сформировать линейно отделимое множество входных векторов, либо использование многослойных персептронов. Можно также применить другие типы нейронных сетей, например линейные сети или сети с обратным распространением, которые могут выполнять классификацию линейно неотделимых векторов входа.

**Пример создания модели персептрона.**

*Синтаксис:*

net = newp(PR,s,tf,lf)

*Описание:*

Персептроны предназначены для решения задач классификации входных векторов, относящихся к классу линейно отделимых.

Функция net = newp(PR, s, tf, lf) формирует нейронную сеть персептрона.

*Входные аргументы:*

PR - массив размера Rx2 минимальных и максимальных значений для R векторов входа;

s - число нейронов;

tf- функция активации из списка {hardlim, hardlims), по умолчанию hardlim;

lf - обучающая функция из списка {learnp, learnpn), по умолчанию learnp.

*Выходные аргументы:*

net - объект класса network object.

*Свойства:*

Персептрон - это однослойная нейронная сеть с функциями взвешивания dotprod, накопления потенциала netsum и выбранной функцией активации. Слой характеризуется матрицей весов и вектором смещений, которые инициализируются М-функцией initzero.

Адаптация и обучение выполняются М-функциями adaptwb и trainwb, которые модифицируют значения весов и смещений до тех пор, пока не будет достигнуто требуемое значение критерия качества обучения в виде средней абсолютной ошибки, вычисляемой М-функцией mае.

*Пример:*

Создать персептрон с одним нейроном, входной вектор которого имеет 2 элемента, значения которых не выходят за пределы диапазона (рис. 2.5):

net = newp ([0 1; 0 1],1);

gensim(net) % Рис.2.5

Определим следующую последовательность двухэлементных векторов входа Р, составленных из 0 и 1:

Р = [[0; 0] [0; 1] [1; 0] [1; 1]];

Обучим персептрон выполнять операцию ЛОГИЧЕСКОЕ И. С этой целью для полного набора входных векторов сформируем последовательность целей:

P1 = cat(2, Р{:});

Т1 = num2cell(P1(1, :) & Pl(2, :))

T1 = [0] [0] [0] [1]



Рис. 2.5.Модель персептрона.

Применим процедуру адаптации, установив число проходов равным 10:

net.adaptParam.passes = 10;

net = adapt(net,P,T1);

Вектор весов и смещение можно определить следующим образом:

net.IW{1}, net.b{1}

ans = 2 1

ans = -3

Таким образом, разделяющая линия имеет вид:

L: 2р1 + р2 - 3 = 0.

Промоделируем спроектированную нейронную сеть, подав входную обучающую последовательность:

Y = sim(net,P)

Y = [0] [0] [0] [1]

Настройка параметров сети выполнена правильно.

Обучим персептрон выполнять операцию НЕИСКЛЮЧАЮЩЕЕ ИЛИ. С этой целью для полного набора входных векторов Р сформируем последовательность целей:

P1 = cat (2, P{:});

Т2 = num2cell(P1(1, :) | Р1(2, :))

Т2 = [0] [1] [1] [1]

Применим процедуру обучения, установив число циклов равным 20:

net.trainParam.epochs =20;

net = train(net,P,T2);

Вектор весов и смещение можно определить следующим образом:

net.IW{1}, net.b{1}

net.IW{1}, net.b{1}

ans = 2 2

ans = -2

Таким образом, разделяющая линия имеет вид:

L:2p1+2p2-2 = 0.

Промоделируем спроектированную нейронную сеть, подав входную обучающую последовательность:

Y = sim(net,P)

Y = [0] [1] [1] [1]

Обучение и настройка сети выполнены правильно.

*Замечание:*

Персептроны решают задачу классификации линейно отделимых входных векторов за конечное время. В случае больших по длине входных векторов функция обучения learnpn может быть по времени выполнения предпочтительнее функции обучения learnp.

*Сопутствующие функции:* SIM, INIT, ADAPT, TRAIN, HARDLIM, HARDLIMS, LEARNP, LEARNPN.

**Задачи:**

1. Построить средствами системы MatLab в специальном окне график прямой с коэффициентами из таблицы вариантов (вариант – номер по списку).
2. Сформировать обучающую выборку из двух множеств точек координатной плоскости, расположенных по разные стороны от построенной разделяющей прямой.
3. Ознакомиться со стандартной моделью персептрона из системы MatLab.
4. Определить однослойный персептрон с двумя входами, используя средства системы MatLab и обучить его на полученных множествах обучающей выборки.
5. Проверить правильность распознавания на обучающих выборках.
6. Решить задачу распознавания с помощью построенного персептрона для 5-6 точек координатной плоскости, не входящих в обучающую выборку.
7. Из обученной модели взять коэффициенты разделяющей прямой (b, w) и построить по ним график новой разделяющей прямой средствами системы MatLab.
8. Сравнить оба графика.
9. Сделать выводы.

**Ход работы**

1. Из таблицы вариантов выбрать два коэффициента для разделяющей прямой:

* W - весовой коэффициент,
* P – коэффициент смещения.

Тогда формула прямой, разделяющей координатную плоскость на части имеет вид:

y=W\*x+p.

Для построения графика этой прямой воспользоваться командой plot и навыками, полученными при выполнении Лабораторной работы 1.

Эта прямая условно разделяет точки декартовой плоскости на 2 группы – можно условно их назвать «черные» и «белые».

1. Выбрать 2 множества точек: 10 «черных» и 10 «белых» и выписать их координаты. Координаты можно выбирать как целые, так и действительные (по желанию).

Из этих двух множеств сформировать обучающую выборку – два массива P и T в среде MatLab в командном окне.

P – двумерный массив, содержит координаты выбранных точек по принципу: первая строка – координаты х (абсциссы), вторая строка – координаты y (ординаты). Точки можно указывать в любом порядке, но помнить, где «черные», а где «белые».

T – одномерный массив, определяет к какому множеству относятся точки. Принято для точек, лежащих выше прямой («белых») указывать 1, а для точек ниже прямой («черных») – указывать 0.

Количество элементов массива T равно количеству столбцов массива P и «цветовая принадлежность» указывается в том же порядке, что и координаты точек в массиве P.

1. Однослойный персептрон Розенбладта в среде MatLab создается командой newp.

С помощью команды help newp вывести на экран подробную информацию о команде newp и внимательно изучить параметры этой команды.

1. Создать однослойный персептрон с помощью команды newp, используя по возможности стандартные параметры, функцию «hardlim» в качестве нелинейного преобразователя. Персептрон должен иметь 2 входа и 1 выход. Имя вашего персептрона задайте как p\_«номер варианта» (например p\_17).

p\_17= newp (…)

Обучить персептрон на сформированных в п. 2 векторах P и T.

Для обучения использовать команду adapt, установив количество шагов обучения (adaptParam.passes) равным 10:

p\_17.adaptParam.passes = 10

p\_17 = adapt(p\_17,P,T)

1. Проверить правильность обучения персептрона на обучающей выборке с помощью команды sim:

Y = sim(p\_17,P)

Если обучение прошло правильно, то полученный вектор Y будет полностью повторять вектор T. Если не совпадает, следует повторить обучение, увеличив количество шагов обучения на 10.

Этот шаг следует повторять до тех пор, пока проверка не выполнится.

1. Выбрать по 5-6 «черных» и «белых» точек на координатной плоскости, не совпадающих с точками из обучающей выборки. Из них сформировать двумерных массив PX по тому же принципу, что и формировали массив P.

Провести распознавание построенным персептроном для этих точек с помощью команды sim.

1. Обученный персептрон представляет собой структуру с разнотипными полями. Среди этих полей есть весовой коэффициент и коэффициент смещения для новой разделяющей прямой. Получить их значения можно так:

p\_17.IW – вектор из двух весовых коэффициентов,

p\_17.net.b – коэффициент смещения.

Уравнение прямой будет иметь вид:

iw(1)\*x+iw(2)\*y+b=0.

Уравнение следует перевести в стандартную форму y=f(x) и построить график полученной прямой средствами системы MatLab.

1. Сравнить полученные графики. Для этого удобно вывести их в одном графическом окне (см. теоретические материалы для Лабораторной работы 1).
2. Сделать выводы о результатах, полученных в процессе выполнения работы.
3. **Дополнительными баллами** будут оценены подробные графические иллюстрации к лабораторной работе: оба графика прямых выведены в одном окне, в том же окне разными цветами (маркерами) обозначены точки для обучающей выборки. В отдельном окне – график прямой для обученного персептрона и точки массива PX, обозначенные разными цветами или маркерами.

**Содержание отчета.**

1. Краткое теоретическое введение.
2. График разделяющей прямой и два набора точек для обучения. Описание процесса их построения
3. Описание создания персептрона, формирования обучающей выборки, процесса его обучения, проверки правильности обучения.
4. Проверка работы персептрона на выбранных точках массива PX.
5. Описания процесса вывода коэффициентов новой разделяющей прямой и получения ее формулы.
6. График новой разделяющей прямой. Описание процесса ее построения.
7. Выводы

**Варианты коэффициентов для решающего правила**

Таблица 2.2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | W | P | **№** | W | P | **№** | W | P | **№** | W | P | **№** | W | P |
| **1** | 10 | -10 | **6** | 5 | -5 | **11** | -9 | 1 | **16** | -4 | 6 | **21** | -8 | 6 |
| **2** | 1 | -9 | **7** | 6 | -4 | **12** | -8 | 2 | **17** | -3 | 7 | **22** | 4 | 2 |
| **3** | 2 | -8 | **8** | 7 | -3 | **13** | -7 | 3 | **18** | -2 | 8 | **23** | -1 | 2 |
| **4** | 3 | -7 | **9** | 8 | -2 | **14** | -6 | 4 | **19** | -1 | 9 | **24** | 1 | -1 |
| **5** | 4 | -6 | **10** | 9 | -1 | **15** | -5 | 5 | **20** | -10 | 10 | **25** | 2 | 0 |

**Контрольные вопросы:**

1. Что такое персептрон и как он устроен?
2. Как происходит обучение персептрона?
3. Что такое пороговая функция и каких видов она бывает?
4. Что такое явление переобучения персептрона?
5. Что такое выпуклая и вогнутая области?
6. Какие типы областей разделяются линейным решателем?
7. Опишите процедуру создания персептрона в MATLAB и ее параметры.
8. Опишите процедуру обучения персептрона в MATLAB и ее параметры.