# Лабораторная работа 4. Процедуры настройки параметров персептронных нейронных сетей. Процедура адаптации

## Цель лабораторных занятий

Изучение алгоритма настройки параметров персептронных нейронных сетей с помощью процедуры адаптации в системе MATLAB.

## Краткие сведения из теории

Многократно используя функции sim и learnp для изменения весов и смещения персептрона, можно в конечном счете построить разделяющую линию, которая решит задачу классификации при условии, что персептрон может решать ее. Каждая реализация процесса настройки с использованием всего обучающего множества называется *проходом* или *циклом*. Такой цикл может быть выполнен с помощью специальной функции адаптации *adapt*. При каждом проходе функция *adapt* использует обучающее множество, вычисляет выход, погрешность и выполняет подстройку параметров персептрона.

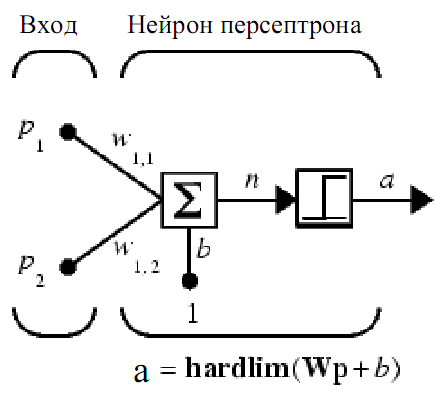
Процедура адаптации не гарантирует, что синтезированная сеть выполнит классификацию нового вектора входа. Возможно, потребуется новая настройка матрицы весов **W** и вектора смещений **b** с использованием функции *adapt*.

Чтобы пояснить процедуру адаптации, рассмотрим простой пример. Выберем персептрон с одним нейроном и двухэлементным вектором входа (рисунок).

Эта сет достаточно проста, так что все расчеты можно выполнить вручную.

Предположим, что можно с помощью персептрона решить задачу классификации векторов, если задано следующее обучающее множество:

(1)



Используем нулевые начальные веса и смещения. Для обозначения переменных каждого шага используем круглые скобки. Таким образом, начальные значения вектора весов **w**Т(0) и смещения *b*(0) соответственно равны **w**Т(0) = [0 0] и *b*(0) = 0.

*1-й шаг процедуры адаптации*

Вычислим выход персептрона для первого вектора входа **p**1, используя начальные веса и смещение:

(2)

Выход не совпадает с целевым значением , поэтому необходимо применить правило настройки (обучения) персептрона, чтобы вычислить требуемые изменения весов и смещений:

(3)

Вычислим новые веса и смещение, используя введенные ранее правила обучения персептрона.

(4)

*2-й шаг процедуры адаптации*

Обратимся к новому вектору входа **p**2, тогда

(5)

В этом случае выход персептрона совпадает с целевым выходом, так что погрешность равна 0 и не требуется изменений в весах или смещении. Таким образом,

(6)

*3-й шаг процедуры адаптации*

Продолжим этот процесс и убедимся, что после третьего шага настройки не изменились:

(7)

*4-й шаг процедуры адаптации*

После четвертого примем значение

(8)

Чтобы определить, получено ли удовлетворительное решение, требуется сделать один проход через все векторы входа с целью проверить, соответствуют ли решения обучающему множеству.

*5-й шаг процедуры адаптации*

Вновь используем первый член обучающей последовательности и получаем

(9)

*6-й шаг процедуры адаптации*

Переходя ко второму члену, получим следующий результат:

(10)

Этим заканчиваются ручные вычисления.

### Расчеты с использованием функции *adapt*.

Вновь сформируем модель персептрона, изображенного на рисунке:

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

Введем первый элемент обучающего множества:

p = {[2; 2]}; t = {0};

Установим параметр passes (число проходов), равным 1, и выполним один шаг настройки:

net.adaptParam.passes = 1;

[net,a,e] = adapt(net,p,t); a,e

a =

[1]

e =

[-1]

Скорректированные вектор весов и смещение определим следующим образом:

twts = net.IW{1,1}, tbiase = net.b{1}

twts =

-2 -2

tbiase =

-1

Это совпадает с результатами, полученными при ручном расчете. Теперь можно ввести второй элемент обучающего множества и т. д., то есть повторить всю процедуру ручного счета и получить те же ре- зультаты.

Но можно эту работу выполнить автоматически, задав сразу все обучающее множество и выполнив один проход:

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

net.trainParam.passes = 1;

p = {[2;2] [1;-2] [-2;2] [-1;1]};

t = {0 1 0 1};

Теперь обучим сеть.

[net,a,e] = adapt(net,p,t);

Возвращаются выход и ошибка

a, e

a =

[1] [1] [0] [0]

e =

[-1] [0] [0] [1]

Скорректированные вектор весов и смещение определяем следующим образом:

twts = net.IW{1,1}, tbiase = net.b{1}

twts =

-3 -1 tbiase =

0

Моделируя полученную сеть по каждому входу, получим

a1 = sim(net,p)

a1 =

[0] [0] [1] [1]

Можно убедиться, что не все выходы равны целевым значениям обучающего множества. Это означает, что следует продолжить на- стройку персептрона.

Выполним еще один цикл настройки:

[net,a,e] = adapt(net,p,t); a, e

a =

[0] [0] [0] [1]

e =

[0] [1] [0] [0]

twts = net.IW{1,1}, tbiase = net.b{1}

twts =

-2 -3 tbiase =

1

a1 = sim(net,p)

a1 =

[0] [1] [0] [1]

Теперь решение совпадает с целевыми выходами обучающего множества, и все входы классифицированы правильно.

Если бы рассчитанные выходы персептрона не совпали с целевыми значениями, то необходимо было бы выполнить еще несколько циклов настройки, применяя функцию adapt и проверяя правильность получаемых результатов.

Итак, для настройки (обучения) персептрона применяется процедура адаптации, которая корректирует параметры персептрона по результатам обработки каждого входного вектора. Применение функции *adapt* гарантирует, что любая задача классификации с линейно отделимыми векторами будет решена за конечное число циклов настройки.

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 векторы входа линейно неотделимы, то процедура адаптации не в состоянии классифицировать все векторы должным образом.

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

Итак, основное назначение персептронов – решать задачи классификации. Они великолепно справляются с задачей классификации линейно отделимых векторов, при этом сходимость гарантируется за конечное число шагов.

Количество циклов обучения зависит от длины отдельных векторов, но и в этом случае решение может быть построено.

Решение задачи классификации линейно неотделимых векторов возможно либо путем предварительной обработки входных векторов с целью сформировать линейное отделимое множество входных векторов, либо путем использования многослойных персептронов. Возможно также применить другие типы нейронных сетей, например, линейные сети или сети с обратным распространением, которые могут выполнять классификацию линейно неотделимых векторов входа.

## Индивидуальное задание

1. Для обучающего множества персептронной нейронной сети, разработанной в лабораторной работе 3, при нулевых начальных значениях весов и смещения выполнить процедуру адаптации ручным расчетом и моделированием с использованием функции *adapt* системы MATLAB.
2. Определить количество циклов настройки сети. Сравнить результаты расчетов с результатами, полученными в лабораторной работе 3.
3. Осуществить моделирование настроенной нейронной сети для пяти новых наборов входных векторов и проверить правильность решения задачи классификации сетью.
4. Повторить процедуру настройки персептронной нейронной сети при нулевых начальных значениях весов и смещения с использованием функции adapt системы MATLAB, увеличив длину одного из векторов обучающего множества в 10–30 раз. Сравнить количество циклов обучения с результатами п. 1.
5. Повторить процедуру настройки персептронной нейронной сети при нулевых начальных значениях весов и смещения с использованием функции *adapt* системы MATLAB, уменьшив длину одного из векторов обучающего множества в 10–15 раз. Сравнить количество циклов обучения с результатами п. 1.

## Содержание отчета

* тема лабораторной работы;
* структурная схема нейронной сети;
* ручной расчет настройки сети:
* текст программы и результаты моделирования;
* выводы.