

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**  
**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ГОМЕЛЬСКИЙ**  
**ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**  
**УНИВЕРСИТЕТ имени П.О.СУХОГО»**

**Кафедра «Информационные технологии»**

**И.А. Мурашко, Д.Е. Храбров**  
**Базы знаний и поддержка принятия решений**

**Лабораторный практикум по одноименной дисциплине**  
**для студентов специальности 1–40 01 02**  
**«Информационные системы и технологии» дневной**  
**формы обучения**

**Гомель 2014**

# Лабораторная работа №1

## «Язык программирования ЛИСП» (8 часов)

### 1.1 Пример простейшей программы на языке ЛИСП

*Целью работы является знакомство с языком Лисп.*

Разработать программу вычисления факториала на процедурном (Pascal, C, можно C#), объектно-ориентированном (C++ или Java) и функциональном языке программирования (Lisp). Найти максимальное значение аргумента, от которого можно вычислить факториал без потери точности (в целых числах).

Пример программы на языке Лисп:

```
(defun fact1 (n)
  (cond ((zerop n) 1)
        (t (* n (fact1 (- n 1))))))
```

Примеры выполнения:

```
> (fact1 5)
```

120

```
> (fact1 25)
```

15511210043330985984000000

```
> (fact1 1000)
```

40238726007709377354370243392300398571937486421071463254379991042993851239  
86290205920442084869694048004799886101971960586316668729948085589013238296  
69944590997424504087073759918823627727188732519779505950995276120874975462  
49704360141827809464649629105639388743788648733711918104582578364784997701  
24766328898359557354325131853239584630755574091142624174743493475534286465  
76611667797396668820291207379143853719588249808126867838374559731746136085  
37953452422158659320192809087829730843139284440328123155861103697680135730  
42161687476096758713483120254785893207671691324484262361314125087802080002  
61683151027341827977704784635868170164365024153691398281264810213092761244  
89635992870511496497541990934222156683257208082133318611681155361583654698  
40467089756029009505376164758477284218896796462449451607653534081989013854  
42487984959953319101723355556602139450399736280750137837615307127761926849  
03435262520001588853514733161170210396817592151090778801939317811419454525  
72238655414610628921879602238389714760885062768629671466746975629112340824  
39208160153780889893964518263243671616762179168909779911903754031274622289  
98800519544441428201218736174599264295658174662830295557029902432415318161  
72104658320367869061172601587835207515162842255402651704833042261439742869  
33061690897968482590125458327168226458066526769958652682272807075781391858  
17888965220816434834482599326604336766017699961283186078838615027946595513  
11565520360939881806121385586003014356945272242063446317974605946825731037

90084024432438465657245014402821885252470935190620929023136493273497565513  
95872055965422874977401141334696271542284586237738753823048386568897646192  
73838149001407673104466402598994902222217659043399018860185665264850617997  
02356193897017860040811889729918311021171229845901641921068884387121855646  
12496079872290851929681937238864261483965738229112312502418664935314397013  
74285319266498753372189406942814341185201580141233448280150513996942901534  
83077644569099073152433278288269864602789864321139083506217095002597389863  
55427719674282224875758676575234422020757363056949882508796892816275384886  
33969099598262809561214509948717012445164612603790293091208890869420285106  
40182154399457156805941872748998094254742173582401063677404595741785160829  
23013535808184009699637252423056085590370062427124341690900415369010593398  
3835777939410970027753472000  
00  
00  
00

>

Предусмотреть возможность проверки корректности вычислений.

## 1.2 Базовые функции языка ЛИСП

*Целью работы является изучение представления данных в языке Лисп в виде атомов, списков, консов и символьных выражений.*

Пускай поставлена задача замены всех положительных элементов простого списка на некоторую константу "z". Реализация такой задачи может выглядеть следующим образом:

```
(defun replace_plus_numbers (mylst)
  (cond
    (
      (null mylst)
      nil
    )
    (
      (plusp (car mylst) )
      (cons "z" (replace_plus_numbers (cdr mylst) ) )
    )
    (
      (cons (car mylst) (replace_plus_numbers (cdr mylst) ) )
    )
  )
)
```

)

Вызывать эту функцию можно так:

```
> (replace_plus_numbers '(1 -2 3 0 -5 9))
```

```
("z" -2 "z" 0 -5 "z")
```

```
> (replace_plus_numbers (list 1 -2 3 0 -5 9))
```

```
("z" -2 "z" 0 -5 "z")
```

Функция `replace_plus_numbers` рекурсивна, единственный её параметр (`mylst`) – список. Тело функции состоит из вызова одной функции `cond`, которая проверяет различные условия. Первое условие (`null mylst`) – терминальная ветвь рекурсии. Если список пуст, то необходимо остановить выполнение. Второе условие – проверка первого элемента списка на положительность (`plusp`). Если первый элемент списка `mylst` положителен, то будет вызван код (`cons "z" ...`), который «склеит» "z" и хвост списка, который рекурсивно будет проверен с помощью определённой функции `replace_plus_numbers`. При вызове в функцию передаётся (`cdr mylst`), т.е. после вызова список будет переопределён.

Если же условие на положительность не истинно, то выполняется код по умолчанию, указанный в конце `cond`. Т.е. вместо склеивания "z" с хвостом выполняется склеивание реального первого элемента списка (`car mylst`) с рекурсивно проверенным хвостом.

Изучаются базовые функции языка для работы со списками – `CAR`, `CDR`, `CONS`, предикаты `ATOM` и `EQ` и условное выражение `COND`. Разрабатываются программы для решения следующих задач:

1. Удалить первый и последний элемент списка.
2. В простом списке чисел заменить все отрицательные числа нулями.
3. Из простого списка чисел удалить все нулевые элементы
4. Продублировать все вхождения атома `X` в данный список.
5. Подсчитать число вхождений атома `X` в простой список.

### 1.3 Числовые функции языка ЛИСП

Необходимо найти максимальный делитель натурального числа `N`. Реализация на языке `Lisp`:

```
(defun max_divisor (N i)
```

```
  (cond
```

```
    (
```

```
      (zerop (rem N i) )
```

```
      i
```

```
)
(
  (max_divisor N (- i 1) )
)
)
)
```

Вызов:

```
> (max_divisor 100 99)
50
> (max_divisor 101 100)
1
> (max_divisor 102 101)
51
```

Параметрами функции являются непосредственно число  $N$  и параметр «цикла», который будет уменьшаться на единицу до тех пор, пока  $N$  нацело не разделится на  $i$ . Подразумевается, что изначально необходимо передавать  $i$  на единицу меньше  $N$ .

Если остаток от деления  $N$  на  $i$  будет равен нулю (`zero? (rem N i)`), то искомый делитель найден. В противном случае рекурсивно вызывается `max_divisor` с  $i$ , уменьшенным на единицу. Если число простое, то оно делится на 1. Терминальная ветвь в данном случае не обязательна.

Целью работы является изучение основных числовых функций языка. Разрабатываются программы для решения следующих задач:

1. Выделить первую цифру натурального  $N$ .
2. Получить все делители натурального  $N$ .
3. Подсчитать число и сумму цифр целого  $N$ .
4. Найти все общие делители натуральных  $M$  и  $N$ .
5. Найти наибольший общий делитель чисел из заданного списка.

## Лабораторная работа №2 «Язык программирования ЛИСП» (8 часов)

### 2.1 Работа со списками в ЛИСПе

Целью работы является закрепление навыков работы со списками. Разрабатываются программы для решения следующих задач:

1. Даны два списка вида  $(a_1, a_2, \dots)$  и  $(b_1, b_2, \dots)$ . Получить список  $(a_1, b_1, a_2, b_2, \dots)$ . Если исходные списки разной длины, то остаток более длинного отбросить.
2. Даны натуральные **m**, **n** и некоторый список. Удалить из списка элементы с номерами с **m**-го по **n**-й.
3. Из произвольного списка удалить все элементы, не являющиеся числом
4. В сложном списке заменить все атомы-числа: положительные на слово “положительное” отрицательные на слово “отрицательное”, нулевые на “ноль”.
5. В произвольном списке удалить элементы с номерами **N** и **M** (без учета скобок)

Пример: Удалить из простого все элементы, начиная с номера **M** и до конца.

Реализация:

```
(defun delFromM (mylst M i)
  (cond
    (
      (null mylst)
      nil
    )
    (
      (> i M)
      nil
    )
    (
      (cons (car mylst) (delFromM (cdr mylst) M (+ 1 i) ))
    )
  )
)
```

Вызов:

```
> (delFromM '(9 2 6 3 7 1) 3 1)
(9 2 6)
```

Функция рекурсивна, передаются: список, **M**, и параметр цикла изначально равный 1. Терминальных ветви две, т.к. во первых необходимо проверить выход за границы, а во

вторых – не включать элементы с номерами, больше *M*. В обычной ситуации выполняется склеивание первого элемента, с его хвостом, обработанным рекурсивно.

## 2.2 Работа со сложными списками в ЛИСПе

Целью работы является закрепление навыков работы со сложными списками.

1. Удалить первый и последний элементы сложного списка (без учета скобок).
2. Произвольный список вида  $(a_1, a_2, \dots, a_k)$  разбить на два подсписка  $(a_1, a_3, a_5 \dots)$  и  $(a_2, a_4, a_6 \dots)$ .
3. Из произвольного списка удалить все отрицательные элементы.
4. Реверсировать произвольный список (включая подсписки).
5. Из произвольного списка удалить все латинские буквы, расположенные между буквами **d** и **k**.

Пример: Заменить все положительные элементы произвольного списка на некоторую константу "z". Реализация такой задачи может выглядеть следующим образом:

```
(defun replace_plus_numbers (mylst)
  (if (null mylst) nil
      (let ((head (car mylst)) (tail (cdr mylst)))
        (cond ((listp head) (cons (replace_plus_numbers head) (replace_plus_numbers tail)))
              ((plusp head) (cons "z" (replace_plus_numbers tail)))
              (t (cons head (replace_plus_numbers tail))))
        )
      )
  )
)
```

Вызов:

```
> (replace_plus_numbers '(1 -2 (3 (-4 5) 6) -7 8 9) )
("z" -2 ("z" (-4 "z") "z") -7 "z" "z")
```

Тело функции начинается с терминальной ветви рекурсии, остальной код выполняется только если список не пуст. Для простоты кода переменной *head* присвоено значение  $(\text{car } \text{mylst})$ , а переменной *tail* – значение  $(\text{cdr } \text{mylst})$ . Основной код заключён в условие *cond*. Если *head* – список, то склеиваем рекурсивно обработанный *head* с рекурсивно обработанным *tail*. Если *head* – положительное число, то склеиваем "z" с рекурсивно обработанным *tail*. Иначе *head* – нечто иное, не обрабатываем его, но склеиваем с обработанным хвостом.

# Лабораторная работа №3

## «Язык программирования ЛИСП» (4 часа)

### Работа с числовыми функциями

Целью работы является закрепление навыков работы с числовыми функциями.

Разрабатываются программы для решения следующих задач:

1. Проверить, является ли список списком чисел, или нет
2. В списке чисел найти значение наименьшего из положительных чисел.
3. Проверить, является ли простой список чисел монотонной последовательностью.
4. Слить два упорядоченных по возрастанию списка чисел в один список.
5. Дан список  $(a_1, a_2, \dots, a_N)$ . Вычислить значение выражения-  
 **$\max(a_2, a_4, \dots) + \min(a_1, a_3, \dots)$** .
6. Дано  $A$  и натуральное  $N$ . Вычислить выражение  $A*(A-N)*(A-2*N)* (A-N^{**2})$ .
7. Дано  $X$  и натуральное  $N$ . Вычислить  $\sin(X)$ , используя разложение в ряд Тейлора. В разложении учитывать  $N$  членов ряда.
8. Вычислить с заданной точностью EPS выражение  $1-1/2+1/3-1/4+ \dots$
9. Методом дихотомии (половинного деления) найти корень уравнения  **$x+\ln(x+0.5)-0.5=0$**  на отрезке  $[0, 2]$  с заданной точностью EPS.
10. Найти произведение всех ненулевых чисел из заданного сложного списка.

Пример: В произвольном списке найти значение максимального из положительных чисел.

```
(defun doFindMax (mylst curMax)
  (if (null mylst) curMax
      (let ((head (car mylst)) (tail (cdr mylst)))
        (cond ((listp head)
               ; Head is a list. Get max from head and tail
               (let ((headmax (doFindMax head curMax)) (tailmax (doFindMax tail curMax)) )
                 (if (> headmax tailmax ) headmax
                     tailmax
                 )
               )
              )
        )
      ((plusp head)
       ; Positive integer
       (if (> head curMax) (doFindMax tail head)
```



Терминальная ветвь рекурсии – проверка, является ли значение текущей дроби меньше точности. Если точность достигнута – то возвращаем 0, который будет прибавлен к общей сумме. Если точность не достигнута, то к текущей дроби ( $\frac{1}{2^i}$ ) прибавляем рекурсивно вычисленную остальную часть, в параметрах которой передаётся текущий шаг, увеличенный на единицу.

## Лабораторная работа №4 «Язык программирования ЛИСП» (8 часов)

### 4.1 Числовые функции языка ЛИСП

Целью работы является закрепление навыков работы с числовыми функциями.

1. Вычислить среднее арифметическое отрицательных чисел произвольного списка.
2. Представить целое  $N$  в виде суммы квадратов двух целых чисел.
3. Представить целое  $N$  в виде суммы квадратов 3-х натуральных чисел.
4. Представить целое  $N$  в виде суммы квадратов 4-х натуральных чисел.
5. Определить предикат, который проверяет, является ли произвольный список монотонной последовательностью чисел, или нет (без учета скобок).

Пример: найти целочисленный квадратный корень от целого положительного числа. Если найти невозможно, вывести -1.

```
(defun mysqrt (N i)
  (if (< (* i i) N) (mysqrt N (+ 1 i))
      (if (= (* i i) N) i
          -1)
  )
)
> (mysqrt 25 1)
5
> (mysqrt 26 1)
-1
> (mysqrt 36 1)
6
```

Линейный рекурсивный алгоритм. Переменная  $i$  перебирается от начального значения и до тех пор, пока  $i^2$  не превысит исходного  $N$ . Если  $i^2$  строго равно  $N$ , то  $i$  является ответом.

### 4.2 Циклы и функционалы в ЛИСПе

Целью работы является закрепление навыков работы с функциями Лиспа.

1. Найти максимальную глубину вложенности произвольного списка.
2. Используя управляющую конструкцию **DO**, вычислить среднее арифметическое чисел  $0.1+0.25+\dots+15.1$ .

3. Используя управляющую конструкцию **DOlist**, вычислить среднее арифметическое и дисперсию чисел из простого списка.
4. Даны два списка вида  $(a_1, a_2, \dots)$  и  $(b_1, b_2, \dots)$ . Используя отображающий функционал MAPx, получить список вида  $(a_1, b_1, a_2, b_2, \dots)$ . Если исходные списки разной длины, то остаток более длинного списка дописать в конец без изменения.
5. Разработать функцию преобразования и вычисления математических выражений в префиксную форму.

Пример: Проверить, все ли элементы списка являются целыми и положительными.

```
(defun listplusp (mylst)
  (setq ispp 1)
  (dolist (i mylst)
    (if (not (plusp i)) (setq ispp 0) )
  )
  (if (eq ispp 1)
    "List is good."
    "NOT plusp list!"
  )
)
```

Вывод:

```
> (listplusp '(1 2 3 4))
"List is good."
> (listplusp '(1 -2 3 4))
"NOT plusp list!"
```

Флаг ispp по умолчанию означает, что список удовлетворяет условию. Если позже в цикле `dolist` будут найдены неподходящие элементы, то значение флага будет изменено.

## **Лабораторная работа №5**

### **«Принятие решений на основе метода Монте-Карло» (4 часа)**

**Целью работы** является изучение эффективности применения метода Монте-Карло для поддержки принятия решений в производственно-экономических задачах.

#### 1. Краткие теоретические сведения

Метод Монте-Карло представляет собой универсальный метод, применяемый для приближенного решения задач самых различных классов: вероятностных и детерминированных, дискретных и непрерывных, задач моделирования и оптимизации и т.д.

Вероятностные задачи – это любые задачи, связанные с анализом случайных явлений (случайных событий, величин, процессов). Детерминированные задачи – это задачи, в постановке которых нет никаких случайных факторов.

Дискретные задачи – это задачи, в которых все анализируемые величины могут принимать значения только из некоторого конечного множества допустимых значений. Непрерывные задачи – это задачи, в которых анализируемые величины могут принимать любые значения из некоторого диапазона (этот диапазон может быть как ограниченным, так и неограниченным).

Задачи моделирования – это задачи, связанные с имитацией некоторого объекта (например, сложной технической системы) или явления (например, процесса развития отрасли) с целью определения его характеристик. Задачи оптимизации – это задачи, в которых требуется выбор лучшего из нескольких возможных вариантов решения.

Большинство практических задач не могут быть однозначно отнесены к одному из приведенных выше классов, а содержат элементы, характерные для нескольких из этих классов.

В данной работе основное внимание уделяется применению метода Монте-Карло для решения детерминированных задач поддержки принятия решений. Решение данных задач методом Монте-Карло основано на многократном случайном выборе вариантов решений и их сравнении. При достаточном количестве испытаний (т.е. вариантов решения, выбранных случайным образом) находится решение, близкое к оптимальному (во многих случаях находится оптимальное решение).

Метод Монте-Карло основан на применении псевдослучайных равномерно распределенных чисел (ПСЧ). Расчет (розыгрыш) ПСЧ выполняется по специальным алгоритмам, позволяющим получать бесконечную последовательность таких чисел. Эти алгоритмы разработаны

таким образом, что ПСЧ всегда принимают значения из диапазона от нуля до единицы. При этом ПСЧ обладают следующими свойствами:

- равномерность: ПСЧ могут принимать значения из любой части диапазона  $(0;1)$  с одинаковой частотой;

- случайность: в последовательности ПСЧ нет какой-либо закономерности;

- независимость: любое из значений ПСЧ не зависит от предыдущих ПСЧ.

Все это позволяет рассматривать ПСЧ как случайные величины, распределенные по равномерному закону в диапазоне от нуля до единицы.

Алгоритмы для получения ПСЧ реализованы практически во всех языках программирования и во многих прикладных программах обработки данных. Например, в языке Паскаль для получения ПСЧ используется функция RANDOM, в табличном процессоре EXCEL – функция СЛЧИС. Одна из возможных реализаций ПСЧ приведена в приложении. ПСЧ применяются для многократного случайного выбора вариантов решения задачи. Из этих вариантов выбирается лучший. Чтобы найти вариант решения, достаточно близкий к оптимальному, обычно требуется выбрать много вариантов решения (от нескольких десятков до тысяч). Поэтому применение метода Монте-Карло возможно только с использованием программных средств.

Во многих случаях данные задачи могут быть решены другими (аналитическими) методами, позволяющими найти точное оптимальное решение. Это могут быть методы линейного программирования, динамического программирования, поиска экстремумов функций многих переменных и т.д. Однако применение точных аналитических методов, как правило, возможно только для задач небольшой размерности (с небольшим количеством ограничений, переменных и т.д.). Для реальных задач применение аналитических методов нередко оказывается невозможным из-за большого объема сложных вычислений. Реализация и применение таких методов (даже с использованием компьютера) оказывается практически невозможным или нецелесообразным. Поэтому во многих случаях применение метода Монте-Карло может оказаться единственно возможным способом решения задачи. Алгоритм поиска решения на основе метода Монте-Карло полностью зависит от конкретной задачи.

Рассмотрим пример. На предприятие поступает некоторое сырье для переработки. Известно, что примерно 15% партий сырья имеют повышенное содержание примесей и требуют дополнительной очистки. Часть сырья подвергается контролю при поступлении на предприятие (входной контроль); остальное сырье поступает на переработку без входного контроля. Затраты, связанные с контролем качества одной партии сырья, составляют 8 руб. Если при контроле обнаруживается повышенное содержание примесей, то партия подвергается очистке; затраты на очистку составляют 40 руб. Если партия с повышенным содержанием примесей не подвергалась входному контролю, то необходимость ее очистки обнаруживается в ходе переработки; в этом случае затраты на очистку составляют 70 руб. Требуется найти, какая часть сырья

должна подвергаться входному контролю, чтобы общие затраты на контроль и очистку были минимальными.

Для решения данной задачи выполним моделирование процесса контроля и очистки сырья при различных долях контролируемых партий сырья: от 0 (входной контроль не выполняется) до 100% (контролируется все сырье) с шагом 5%. Предположим, что доля контролируемых партий сырья равна некоторой заданной величине, например, 5%. Смоделируем процесс контроля и очистки большого количества партий сырья (например, десяти тысяч) и определим величину затрат на эти операции. Обозначим долю контролируемых партий как  $P$  (в данном случае  $P=0,05$ ). Моделирование осуществляется по следующему алгоритму.

1. Разыгрываются два ПСЧ:  $R1$  и  $R2$ . Величина  $R1$  будет использоваться для имитации отбора контролируемых партий, а  $R2$  – для имитации наличия (или отсутствия) повышенного содержания примесей.

2. Если  $R1 < P$ , то будем считать, что смоделировано поступление партии сырья на контроль. Общие затраты при этом увеличиваются на 8 руб. (это величина затрат на контроль одной партии). Если при этом  $R2 < 0,15$  (где 0,15 – доля партий, имеющих повышенное содержание примесей), то будем считать, что смоделировано наличие повышенного содержания примесей в проверяемой партии. В этом случае общие затраты увеличиваются на 40 руб. Если  $R1 > P$  (партия сырья не поступает на контроль), и при этом  $R2 < 0,15$  (партия имеет повышенное содержание примесей), то смоделировано обнаружение повышенного содержания примесей при переработке сырья. Затраты увеличиваются на 70 руб.

Шаги 1 и 2 повторяются многократно, например, 10000 раз (это означает моделирование контроля и очистки 10000 партий сырья). Подсчитываются общие затраты на контроль и очистку. В таблице 1 приведен пример десяти испытаний (моделирование контроля и очистки десяти партий).

Таблица 1

Номер испытания	$R1$	Контроль	$R2$	Повышенное содержание примеси	Затраты на данную партию	Общие затраты
1	0,0795	да	0,3780	нет	8	8
2	0,0593	да	0,7602	нет	8	16
3	0,2847	нет	0,8197	нет	0	16
4	0,6133	нет	0,5766	нет	0	16
5	0,9595	нет	0,0981	да	70	86
6	0,2410	нет	0,5962	нет	0	86
7	0,2978	нет	0,6458	нет	0	86
8	0,9762	нет	0,0523	да	70	156
9	0,4523	нет	0,8153	нет	0	156
10	0,4286	нет	0,8400	нет	0	156

Аналогично следует выполнить моделирование при других долях контролируемых партий сырья (от 0 до 100% с шагом 5%).

Приведем программную реализацию данного алгоритма на языке Паскаль. Основные константы и переменные программы:

n – количество испытаний (количество обрабатываемых партий);  
zatr\_cont – затраты на контроль одной партии сырья;

zatr\_o1 и zatr\_o2 – затраты на очистку партии при обнаружении повышенного содержания примеси в ходе контроля и в процессе переработки соответственно;

primes – доля партий сырья, имеющих повышенное содержание примеси;

p – доля контролируемых партий;

proc – доля контролируемых партий сырья, выраженная в процентах (например, если proc=5, то p=0,05);

r1,r2 – ПСЧ;

sum\_zatr – общие затраты на контроль и очистку всех партий.

```
program control;
uses crt;
const n=10000;
zatr_cont=8;
zatr_o1=40;
zatr_o2=70;
primes=0.15;
var p,r1,r2,sum_zatr: real;
proc,i: integer;

begin
clrscr;
proc:=0;
while proc<=100 do
begin
p:=proc/100;
sum_zatr:=0;
for i:=1 to n do
begin
r1:=random; r2:=random;
if r1<p then sum_zatr:=sum_zatr+zatr_cont;
if (r1<p) and (r2<primes) then sum_zatr:=sum_zatr+zatr_o1;
if (r1>=p) and (r2<primes) then sum_zatr:=sum_zatr+zatr_o2;
end;
writeln('Процент контроля: ',proc, ' Затраты на 10000 партий: ',sum_zatr:10:2);
proc:=proc+5;
end;
end.
```

Результаты моделирования процесса контроля и очистки сырья приведены в таблице 2.

Таблица 2

Процент контроля	Затраты на 10000 партий
0	112280
10	105432
20	109976
30	115378
40	121354
50	118078
60	126554
70	129020
80	132382
90	137690
100	139120

Таким образом, предприятию следует подвергать входному контролю 10% партий сырья. При этом затраты на контроль и очистку сырья будут минимальными.



### Задание

На предприятие радиоэлектронной промышленности поступают комплектующие изделия – резисторы с номиналом невысокой точности (15%). Известно, что примерно  $A\%$  резисторов не подходит для изготовления продукции и требуют дополнительной подгонки. Чтобы выявить такие резисторы, необходим входной контроль. Стоимость контроля одного резистора составляет  $B$  руб. Стоимость подгонки составляет  $C$  руб. Если резистор установили в изделие, то стоимость его замены составляет  $D$  руб. Требуется найти, какую часть резисторов необходимо подвергнуть входному контролю, чтобы общие затраты на контроль и подгонку были минимальными.

Псевдослучайные числа формируются на основе LFSR с заданным полиномом.

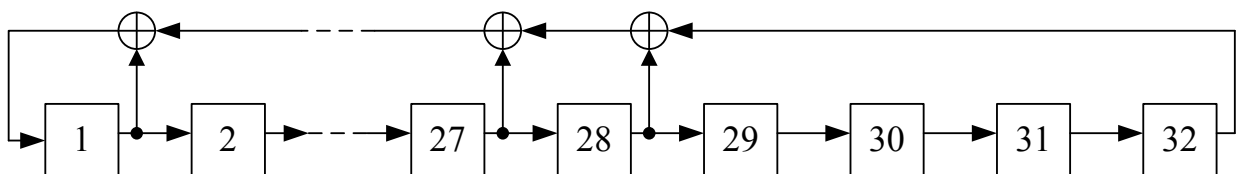
Таблица 3 – Варианты задания

Вариант	$A, \%$	$B$	$C$	$D$	№ полинома
1	5	2	50	150	5
2	5	3	55	145	7
3	10	4	60	140	9
4	10	5	65	135	11
5	15	6	70	130	13
6	15	7	50	125	15
7	20	8	55	120	2
8	20	9	60	115	4
9	25	2	65	110	6
10	25	3	70	105	8
11	7	4	50	100	10
12	7	5	55	95	12
13	17	6	60	90	14
14	17	7	65	85	1
15	19	8	70	80	3

## Варианты полиномов

1.  $X^{32}+X^{30}+X^{28}+X^{27}+X^{25}+X^{22}+X^{21}+X^{20}+X^{19}+X^{18}+X^{12}+X^{10}+X^9+X^6+1$
2.  $X^{32}+X^{30}+X^{29}+X^{26}+X^{25}+X^{24}+X^{23}+X^{20}+X^{18}+X^9+X^5+X^1+1$
3.  $X^{32}+X^{28}+X^{27}+X^{26}+X^{23}+X^{22}+X^{21}+X^{20}+X^{19}+X^{18}+X^{17}+X^{12}+X^9+X^5+X^4+X^2+1$
4.  $X^{32}+X^{30}+X^{28}+X^{25}+X^{24}+X^{23}+X^{21}+X^{20}+X^{19}+X^{18}+X^{17}+X^{13}+X^{11}+X^9+X^5+X^4+X^3+X^1+1$
5.  $X^{32}+X^{31}+X^{30}+X^{29}+X^{27}+X^{26}+X^{25}+X^{24}+X^{23}+X^{21}+X^{20}+X^{17}+X^{16}+X^{14}+X^{12}+X^9+X^8+X^7+1$
6.  $X^{32}+X^{31}+X^{30}+X^{29}+X^{27}+X^{25}+X^{23}+X^{21}+X^{20}+X^{19}+X^{13}+X^{10}+X^7+X^6+X^5+X^4+X^3+X^1+1$
7.  $X^{32}+X^{30}+X^{28}+X^{27}+X^{25}+X^{24}+X^{23}+X^{21}+X^{20}+X^{19}+X^{18}+X^{16}+X^{15}+X^{14}+X^8+X^6+X^5+X^3+1$
8.  $X^{32}+X^{29}+X^{27}+X^{26}+X^{25}+X^{23}+X^{21}+X^{17}+X^{16}+X^{15}+X^{13}+X^{12}+X^8+X^5+X^4+X^3+1$
9.  $X^{32}+X^{31}+X^{27}+X^{26}+X^{23}+X^{22}+X^{19}+X^{18}+X^{17}+X^{16}+X^{15}+X^{12}+X^9+X^8+X^6+X^5+X^4+X^3+X^2+X^1+1$
10.  $X^{32}+X^{31}+X^{30}+X^{29}+X^{27}+X^{24}+X^{23}+X^{22}+X^{18}+X^9+X^7+X^6+X^3+X^2+1$
11.  $X^{32}+X^{31}+X^{30}+X^{28}+X^{27}+X^{24}+X^{23}+X^{21}+X^{18}+X^{16}+X^{15}+X^{13}+X^{12}+X^{11}+X^{10}+X^9+X^7+X^6+X^5+X^2+1$
12.  $X^{32}+X^{29}+X^{28}+X^{23}+X^{22}+X^{21}+X^{16}+X^{13}+X^{10}+X^6+X^5+X^4+1$
13.  $X^{32}+X^{31}+X^{26}+X^{25}+X^{24}+X^{22}+X^{21}+X^{20}+X^{19}+X^{16}+X^{13}+X^{12}+X^8+X^2+1$
14.  $X^{32}+X^{28}+X^{22}+X^{21}+X^{20}+X^{19}+X^{18}+X^{16}+X^{10}+X^8+X^7+X^3+1$
15.  $X^{32}+X^{30}+X^{27}+X^{26}+X^{25}+X^{23}+X^{22}+X^{21}+X^{20}+X^{19}+X^{18}+X^{17}+X^{13}+X^6+X^5+X^4+1$

Пример LFSR с порождающим полиномом  $X^{32}+X^{28}+X^{27}+X^1+1$



**Лабораторная работа №6 + практическое занятие 1**  
**(8 часов + 4 часа)**  
**«Методы принятия решений при многих критериях»**

**Цель работы:** Изучение эффективности применения метода Электра для принятия решений в производственно-технологических задачах.

## **1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

### **1.1 Особенности принятия решений**

В общем случае в процессе принятия решений могут участвовать три категории лиц:

- лицо, принимающее решение (ЛПР), которое несет ответственность за это решение;
- системные аналитики, или исследователи, занимающиеся подготовкой и обоснованием решений;
- эксперты, или высококвалифицированные специалисты, имеющие знания, опыт и интуицию и привлекаемые по отдельным аспектам решаемой задачи.

Потенциально возможные варианты действий называются *альтернативами*, а характеристики качества альтернатив – *критериями*. При решении сложных системных задач рассматривают множество альтернатив (минимум две) и множество критериев, причем учет многих критериев приближает постановку задачи к реальной действительности. Оценка множества альтернатив по множеству критериев происходит с учетом внешних условий.

Принятие решений как научное направление характеризуется тем, что его практически невозможно полностью формализовать, так как в процессе принятия решений используются процедуры многокритериального анализа альтернатив в условиях неполной информации о внешних факторах, влияющих на эффективность решений, учитывается субъективное мнение экспертов и предпочтения ЛПР. Отметим основные проблемы, возникающие при многокритериальной оптимизации.

1) *Противоречивость критериев*. Как правило, улучшение по одному критерию приводит к ухудшению по другому критерию. Например, более производительный компьютер стоит дороже.

2) При многокритериальной оптимизации используются разнородные оценки: числовые, качественные («быстрый» процессор, «отличная машина» и т.п.), оценки вида «да – нет», балльные оценки, оценки в виде ранжирования и т.д., которые к тому же отличаются по размерности (например, цена – в рублях, производительность компьютера – в миллионах операций в секунду, объем оперативной памяти – в гигабайтах и т.п.).

3) Критерии могут сильно различаться по важности.

4) Крайне сложно (а иногда и невозможно) определить аналитическую зависимость между критериями.

## 1.2 Приведение оценок к единой форме

Основные способы приведения оценок.

1) Перевод качественных оценок в числовую форму осуществляется по шкале Харрингтона (таблица 1).

Таблица 1 – Шкала Харрингтона

Качественная оценка	«Очень плохо»	«Плохо»	«Удовлетворительно»	«Хорошо»	«Отлично»
Числовая оценка	0,0 – 0,2	0,2 – 0,36	0,36 – 0,63	0,63 – 0,8	0,8 – 1,0

В случае, если две альтернативы имеют одинаковую оценку «хорошо», но по мнению эксперта или ЛПР вторая альтернатива лучше, то первой альтернативе можно назначить оценку 0,7, а второй – 0,8.

Для оценок, имеющих вид «да – нет», используют следующую шкалу:

«да» – 0,67;

«нет» – 0,33.

2) Оценки различного рода, представленные различными системами измерений, заменяются экспертными оценками, представленными в виде балльных оценок, в долях единицы, в виде парных сравнений, в виде ранжирования и т.д.

3) Нормирование критериев. Числовые оценки из произвольного диапазона приводят к единому масштабу, то есть оценкам, лежащим в диапазоне [0;1]. При этом, как правило, лучшей оценке соответствует большее значение:

$$\bar{c}_i = \frac{c_i}{c_i^*}, \text{ где } c_i^* = \max c_i, i = \overline{1, n}, n - \text{число критериев.}$$

## 1.3 Методы экспертного анализа

### 1.3.1 Попарное сравнение альтернатив

При решении сложных, комплексных проблем, особенно в условиях неопределенности и неполноты информации, широко применяются методы экспертного анализа. Идея экспертного оценивания состоит в том, что для получения оценок привлекаются компетентные в данной области люди – эксперты, которые проводят интуитивно-логический анализ какого-либо вопроса с целью вынесения по нему суждения. Суждения экспертов определенным образом обрабатываются с использованием специальных математических процедур. В результате получают так называемые экспертные оценки. Следует отметить, что экспертная оценка не является решением. Это лишь информация, помогающая ЛПР выработать оптимальное решение. В общем случае предпочтения ЛПР могут не совпадать с предпочтениями экспертов. Однако суждения экспертов, их советы помогают ЛПР критически осмыслить различные точки зрения, уточнить или изменить свою систему предпочтений и тем самым уменьшить вероятность принятия неоптимальных решений.

При использовании этих методов для каждой пары альтернатив определяется оценка превосходства одной альтернативы над другой; эта оценка может непосредственно указываться человеком или вычисляться на основе оценок альтернатив по отдельным критериям. На основе этих сравнений определяется лучшая альтернатива.

Метод парных сравнений основан на попарном сравнении альтернатив. Для каждой пары альтернатив эксперт указывает, какая из альтернатив предпочтительнее (лучше, важнее и т.д.). Существует ряд алгоритмов, реализующих метод парных сравнений: они различаются по количеству используемых экспертных оценок (индивидуальные и коллективные оценки), по шкалам сравнения альтернатив и т.д. Рассмотрим некоторые алгоритмы, реализующие метод парных сравнений.

### 1.3.2 Алгоритм Саати

Алгоритм основан на попарном сравнении альтернатив, выполняемом одним экспертом. Для каждой пары альтернатив эксперт указывает, в какой степени одна из них предпочтительнее другой, по следующему правилу:

-  $x_{ij} = 1$  –  $i$ -я альтернатива равна  $j$ -й;

-  $x_{ij} = 3$  –  $i$ -я альтернатива немного лучше  $j$ -й (если  $j$ -я альтернатива немного лучше  $i$ -й, то  $x_{ij} = 1/3$ );

-  $x_{ij} = 5$  –  $i$ -я альтернатива лучше  $j$ -й (если наоборот, то  $x_{ij} = 1/5$ );

-  $x_{ij} = 7$  –  $i$ -я альтернатива значительно лучше равна  $j$ -й (если наоборот, то  $x_{ij} = 1/7$ );

-  $x_{ij} = 9$  –  $i$ -я альтернатива намного лучше  $j$ -й (если наоборот, то  $x_{ij} = 1/9$ ).

Допускается использование промежуточных оценок 2, 4, 6, 8 и 1/2, 1/4, 1/6, 1/8, соответственно. В результате формируется матрица парных сравнений размером  $n \times n$ , на главной диагонали которой расположены единицы. Пример матрицы приведен в таблице 2.

Таблица 2 – Матрица парных сравнений

Альтернатива	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$c_i$
$X_1$	1	1/3	3	1/5	2	0,83
$X_2$	3	1	5	1/3	5	1,9
$X_3$	1/3	1/5	1	1/7	1	0,39
$X_4$	5	3	7	1	8	3,84
$X_5$	1/2	1/5	1	1/8	1	0,41

Далее находятся цены альтернатив:

$$c_i = \sqrt[n]{\prod_{j=1}^n x_{ij}}, \quad (1)$$

где  $i = \overline{1, n}$ .

Значения цен альтернатив представлено в последнем столбце таблицы.

Находится сумма цен альтернатив:

$$C = \sum_{i=1}^n c_i. \quad (2)$$

В примере  $C = 7,37$ . Затем определяются веса альтернатив:

$$v_i = \frac{c_i}{C}, \quad (3)$$

где  $i = \overline{1, n}$ .

Самой лучшей считается альтернатива, имеющая максимальный вес. Для примера:  $v_1 = 0,11$ ,  $v_2 = 0,26$ ;  $v_3 = 0,05$ ;  $v_4 = 0,52$ ;  $v_5 = 0,06$ . Получили, что по мнению эксперта лучшей считается четвертая альтернатива.

Алгоритм Саати позволяет проводить проверку экспертных оценок на непротиворечивость (то есть выявить ошибки, допущенные экспертом при заполнении матрицы парных сравнений). Для этого находятся суммы столбцов матрицы парных сравнений:

$$r_j = \sum_{i=1}^n x_{ij}, \quad (4)$$

где  $j = \overline{1, n}$ .

Для примера  $r_1 = 9,8$ ,  $r_2 = 4,7$ ,  $r_3 = 17$ ,  $r_4 = 1,8$ ,  $r_5 = 17$ . Рассчитывается вспомогательная величина  $\lambda$ :

$$\lambda = \sum_{j=1}^n r_j v_j. \quad (5)$$

Для примера  $\lambda = 5,1$ . Находится индекс согласованности

$$\upsilon = \frac{\lambda - n}{n - 1}. \quad (6)$$

Для примера  $\upsilon = 0,025$ . В зависимости от размерности матрицы парных сравнений находится величина случайной согласованности  $\gamma$  (таблица 3). Для примера:  $\gamma = 1,12$ .

Таблица 3 – Величина случайной согласованности  $\gamma$

$n$	3	4	5	6	7	8	9	10
$\gamma$	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49

Затем находится отношение согласованности:

$$\mu = \frac{\upsilon}{\gamma}. \quad (7)$$

Если это отношение превышает 0,2, то требуется уточнение матрицы парных сравнений. В данном примере  $\mu = 0,022$ , поэтому уточнение не требуется.

### 1.3.3 Метод ранга

Метод основан на балльных оценках альтернатив, указываемых несколькими экспертами. Каждый из экспертов, независимо от других, оценивает альтернативы по некоторой шкале (пятибалльной, десятибалльной и т.п.). Наиболее предпочтительная альтернатива получает максимальный балл. Эти оценки  $x_{ij}$  сводятся в матрицу размером  $m \times n$ , где  $i = \overline{1, m}$ ,  $m$  – число экспертов,  $j = \overline{1, n}$ ,  $n$  – число альтернатив. Находятся суммарные оценки альтернатив всеми экспертами:

$$c_j = \sum_{i=1}^m x_{ij}. \quad (8)$$

Находится сумма всех оценок  $C = \sum_{j=1}^n c_j$  и веса альтернатив  $v_i = \frac{c_i}{C}$ , где

$i = \overline{1, n}$ . Наиболее предпочтительной является альтернатива, имеющая максимальный вес.

## 1.4 Методы выбора лучших альтернатив

### 1.4.1 Выбор множества Парето-оптимальных решений

Множество Парето представляет собой множество альтернатив, обладающих следующим свойством: любая из альтернатив, входящих во множество Парето, хотя бы по одному критерию лучше любой другой альтернативы, входящей в это множество.

Выбор множества Парето-оптимальных решений (множества Парето) представляет собой отбор перспективных альтернатив, из которых затем отбирается одна (лучшая) альтернатива. Применение принципа Эджворта – Парето позволяет из множества всех возможных исключить заведомо неприемлемые решения, т.е. те, которые никогда не могут оказаться выбранными, если выбор осуществляется достаточно «разумно». После такого исключения остается множество, которое называют множеством Парето или областью компромиссов. Оно, как правило, является достаточно широким, и в процессе принятия решений неизбежно встает вопрос о том, какое именно возможное решение выбрать среди Парето-оптимальных? Выражаясь иначе, какие из Парето-оптимальных решений следует удалить для того, чтобы произвести дальнейшее сужение области компромиссов и, тем самым, получить более точное представление об искомом множестве выбираемых решений? Этот вопрос при решении практических многокритериальных задач является наиболее трудным и наименее проработанным к настоящему времени.

Выбор множества Парето производится следующим образом. Все альтернативы попарно сравниваются друг с другом по всем критериям. Если при сравнении каких-либо альтернатив (обозначим их как  $Y_i$  и  $Y_j$ ) оказывается, что одна из них не лучше другой ни по одному критерию, то ее можно исключить из рассмотрения.

Представим рассмотренную процедуру отбора в виде алгоритма. Пусть множество возможных альтернатив  $Y$  состоит из конечного числа альтернатив и имеет вид:  $Y = \{y^1, y^2, \dots, y^N\}$ . Необходимо сформировать новое множество альтернатив  $P(Y)$  путем исключения альтернатив, которые не превосходят ни по одному критерию остальные альтернативы.

*Шаг\_1.* Вначале положим  $P(Y) = Y$ ,  $i = 1$ ,  $j = 2$ . Т.е. текущее множество парето-оптимальных альтернатив совпадает с исходным множеством.

*Шаг\_2.* Проверяем неравенства  $y^i \geq y^j$  по всем критериям. Если оно истинно, то удаляем из  $P(Y)$  альтернативу  $y^j$  и переходим к *Шагу\_4*. В противном случае перейти к *Шагу\_3*.

*Шаг\_3.* Проверяем неравенства  $y^j \geq y^i$  по всем критериям. Если оно истинно, то удаляем из  $P(Y)$  альтернативу  $y^i$  и переходим к *Шагу\_4*. В противном случае перейти к *Шагу\_4*.



*Шаг\_4.* Если  $j < N$  то  $j = j + 1$  и вернуться на *Шаг\_2*. В противном случае перейти к *Шагу\_5*.

*Шаг\_5.* Если  $i < N - 1$  то  $i = i + 1$ ,  $j = i + 1$  и вернуться на *Шаг\_2*. В противном случае закончить вычисления, так как множество Парето-оптимальных альтернатив сформировано.

Как правило, во множество Парето входит несколько альтернатив. Поэтому выбор множества Парето не обеспечивает принятия окончательного решения (выбора одной лучшей альтернативы), однако позволяет сократить количество рассматриваемых альтернатив, т.е. упрощает принятие решения.

### 1.4.2 Метод ЭЛЕКТРА

Метод ЭЛЕКТРА (ELECTRE – *Elimination Et Choice Traduisant la Realite* – исключение и выбор, отражающие реальность) предложена профессором Б. Руа (Франция). Метод предназначен для решения задач, в которых из имеющегося множества альтернатив требуется выбрать заданное количество лучших альтернатив с учетом их оценок по нескольким критериям, а также важности этих критериев. В зависимости от того, каким образом вычисляется важность критериев, метод ЭЛЕКТРА имеет три модификации – ЭЛЕКТРА I, ЭЛЕКТРА II, ЭЛЕКТРА III.

Основная идея метода заключается в следующем. Для каждой пары альтернатив ( $X_i$  и  $X_j$ ) выдвигается предположение (гипотеза) о том, что альтернатива  $X_i$  лучше, чем  $X_j$ . Затем для каждой пары альтернатив находятся два индекса: индекс согласия (величина, подтверждающая предположение о превосходстве  $X_i$  над  $X_j$ ) и индекс несогласия (величина, опровергающая это предположение). На основе анализа этих индексов выбирается одна или несколько лучших альтернатив («ядро» альтернатив).

Алгоритм метода состоит из следующих шагов (ЭЛЕКТРА I).

*Шаг\_1.* Выполняется переход к безразмерному виду критериев.

*Шаг\_2.* Определяются индексы согласия  $c_{jk}$ :

$$c_{jk} = \sum_{i \in I^+} v_i,$$

где  $j = \overline{1, n}$ ,  $k = \overline{1, n}$ ,  $v_i$  – вес  $i$ -го критерия,  $0 \leq v_i \leq 1$ ,  $n$  – количество альтернатив,  $I^+$  – подмножество критериев, по которым  $j$ -я альтернатива не хуже  $k$ -й.

*Шаг\_3.* Определяются индексы несогласия  $d_{jk}$ :

$$d_{jk} = \max_{i \in I^-} (p_{ik} - p_{ij}),$$

где  $j = \overline{1, n}$ ,  $k = \overline{1, n}$ ,  $p_{ik}, p_{ij}$  – безразмерные оценки,  $n$  – количество альтернатив,  $I^-$  – подмножество критериев, по которым  $j$ -я альтернатива не превосходит  $k$ -ю.

*Шаг\_4.* Находится предельное значение индекса согласия:

$$c_j = \min_k c_{jk},$$

где  $j = \overline{1, n}$ .

*Шаг\_5.* Находится предельное значение индекса несогласия:

$$d_j = \max_k d_{jk},$$

где  $j = \overline{1, n}$ .

*Шаг\_6.* Выделяются лучшие альтернативы или «ядро» альтернатив:

$$c_j > c^*,$$

$$d_j < d^*,$$

где  $c^*$ ,  $d^*$  – пороговые значения индексов согласия и несогласия, соответственно (как правило, используют  $c^* = d^* = 0,5$ ).

В методе ЭЛЕКТРА II индексы согласия подсчитываются тем же способом, что и в методе ЭЛЕКТРА I. Отличием является то, что в нем задаются два уровня для индекса согласия  $\alpha_1 > \alpha_2$  и два уровня индекса несогласия  $\beta_1 \leq \beta_2$ , а так же вводятся два отношения предпочтения  $\delta_1$  и  $\delta_2$  между альтернативами.

В методе ЭЛЕКТРА III используются псевдокритерии и числовые бинарные отношения.

## **2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

1. Изучить основные особенности задач выбора альтернатив по многим критериям, виды и основные возможности методов их решения.

2. Решить задачу выбора оптимального решения по многим критериям.

Задача решается в следующем порядке:

– выбрать множество Парето;

– вычислить веса критериев, используя методы экспертного анализа (метод ранга);

– выполнить анализ отобранных альтернатив по методу ЭЛЕКТРА.

Предусмотреть возможность выбора одной или двух альтернатив, изменяя пороговые значения индексов согласия и несогласия.

## Задание 1

Предприятие – производитель изделий бытовой электроники выбирает торговую фирму для заключения с ней договора о распространении своей продукции. Имеется шесть торговых фирм, о которых известно следующее.

Фирма	ТФ1	ТФ2	ТФ3	ТФ4	ТФ5	ТФ6
Опыт работы с данной продукцией, лет	5	2	6	5	7	4
Торговая сеть	развитая	развитая	развитая	средняя	средняя (немного хуже, чем у ТФ4)	средняя (немного лучше, чем у ТФ4 и ТФ5)
Репутация	сомнительная	хорошая	средняя	хорошая	средняя	хорошая

Важность критериев оценивается двумя экспертами.

По мнению первого эксперта, основной критерий – репутация, менее важный – опыт работы, еще менее важный – торговая сеть.

По мнению второго эксперта, основной критерий – репутация, менее важный – торговая сеть, еще менее важный – опыт работы.

## Задание 2

Предприятие предполагает приобрести станок. Характеристики станков, из которых делается выбор, следующие.

Станок	СТ1	СТ2	СТ3	СТ4	СТ5	СТ6
Производительность, изделий/час	25	25	30	15	20	35
Стоимость станка, тыс. у.е.	140	100	200	100	100	200
Надежность	достаточно высокая	средняя	очень высокая	достаточно высокая (немного ниже, чем у СТ1 и СТ6)	средняя	достаточно высокая

Важность критериев оценивается двумя экспертами.

По мнению первого эксперта, основной критерий – производительность, немного менее важный – надежность, еще немного менее важный – стоимость.

По мнению второго эксперта, основной критерий – производительность, менее важный – стоимость, еще немного менее важный – надежность.

### Задание 3

Предлагаются шесть вариантов площадки для строительства нового предприятия. Характеристики площадок следующие.

Площадка	Пл1	Пл2	Пл3	Пл4	Пл5	Пл6
Дорожная сеть	средняя	плохая	развитая	развитая (немного лучше, чем для Пл3)	средняя	плохая
Энергоснабжение	хорошее	хорошее	плохое	среднее	очень хорошее	среднее
Затраты на подготовку к строительству, млн ден.ед.	3,5	2,5	3	3,5	3	2,0

Важность критериев оценивается двумя экспертами.

По мнению первого эксперта, наиболее важный критерий – затраты на подготовку к строительству, менее важны (и одинаково важны между собой) дорожная сеть и энергоснабжение.

По мнению второго эксперта, наиболее важный критерий – дорожная сеть, немного менее важный – затраты на подготовку к строительству, еще немного менее важный – энергоснабжение.

### Задание 4

Предлагаются шесть проектов строительства промышленного предприятия. Характеристики проектов следующие.

Проект	П1	П2	П3	П4	П5	П6
Прибыль, млн ден.ед./год	12	10	13	11	15	14
Новые рабочие места	3000	3500	3000	1500	2000	2500
Возможности развития территории	хорошие	средние	средние (немного хуже, чем для П2)	хорошие	очень хорошие	очень хорошие

Оценка важности критериев выполняется двумя экспертами.

По мнению первого эксперта, наиболее важный критерий – прибыль, менее важный – возможности развития территории, еще менее важный – количество новых рабочих мест.

По мнению второго эксперта, наиболее важный критерий – прибыль, менее важный – количество новых рабочих мест, еще менее важный – возможности развития территории.

### Задание 5

Выбирается место для строительства металлургического предприятия. Характеристики мест, предлагаемых для строительства следующие.

Место	М1	М2	М3	М4	М5	М6
Близость к источникам сырья	совсем близко	близко	далеко	совсем близко	близко (немного дальше, чем для М2)	среднее расстояние
Близость к потребителям	далеко	среднее расстояние	близко	очень далеко	далеко	совсем близко
Затраты на подготовку к строительству, млн ден.ед.	2,5	4	3	2	3	3,5

Важность критериев оценивается двумя экспертами.

По мнению первого эксперта, наиболее важный критерий – затраты на подготовку к строительству; менее важный – близость к источникам сырья, еще немного менее важный – близость к потребителям.

По мнению второго эксперта, наиболее важный критерий – близость к источникам сырья, немного менее важный – затраты на подготовку к строительству, значительно менее важный – близость к потребителям.

### Задание 6

Предприятие предполагает заключить договор о поставках железной руды с одним из шести поставщиков. Характеристики поставщиков следующие.

Поставщик	П1	П2	П3	П4	П5	П6
Содержание металла в руде, %	12	9	15	8	7	10
Стоимость руды, ден.ед./тонну	200	120	220	120	100	140
Надежность поставок	возможны нарушения	высокая	очень высокая	достаточно высокая	высокая	высокая

Важность критериев оценивается двумя экспертами.

По мнению первого эксперта, наиболее важный критерий – содержание металла в руде, следующий по важности – надежность поставок, следующий по важности – стоимость руды.

По мнению второго эксперта, наиболее важный критерий – содержание металла в руде, следующий по важности – стоимость руды, следующий по важности – надежность поставок.

## Задание 7

Предприятие предполагает закупить универсальный станок для изготовления изделий нескольких типов. Характеристики станков, из которых делается выбор, следующие.

Станок	СТ1	СТ2	СТ3	СТ4	СТ5	СТ6
Количество типов выпускаемых изделий	10	12	8	15	10	12
Стоимость станка, тыс. ден.ед.	200	250	160	250	180	240
Удобство переналадки на другой тип изделия	достаточно простая	достаточно простая	сложная	очень простая	сложная (немного сложнее, чем для СТ3)	достаточно простая

Важность критериев оценивается двумя экспертами.

По мнению первого эксперта, наиболее важный критерий – количество типов выпускаемых изделий, менее важный – стоимость, еще менее важный – удобство переналадки.

По мнению второго эксперта, наиболее важный критерий – стоимость, менее важный – удобство переналадки, еще менее важный – количество типов выпускаемых изделий.

## Задание 8

Предлагаются шесть вариантов площадки для строительства нового предприятия химической промышленности. Характеристики площадок следующие.

Площадка	Пл1	Пл2	Пл3	Пл4	Пл5	Пл6
Условия для доставки сырья	хорошие	отличные	средние	хорошие (немного хуже, чем для Пл1)	средние	очень хорошие
Затраты на подготовку к строительству, млн ден.ед.	3,5	1,8	4	3	3,5	4
Опасность загрязнения грунтовых вод в случае аварии	загрязнение возможно	высокая опасность	опасность мала	загрязнение возможно	опасность мала	опасность мала

Важность критериев оценивается двумя экспертами.

По мнению первого эксперта, наиболее важный критерий – опасность загрязнения, немного менее важный – затраты на подготовку к строительству, еще немного менее важный – условия для доставки сырья.

По мнению второго эксперта, наиболее важный критерий – затраты на подготовку к строительству, примерно такой же по важности (немного менее важный) – опасность загрязнения, менее важный – условия для доставки сырья.

# **Лабораторная работа №7 + Практическое занятие 2**

## **(8 часов + 4 часа)**

### **«Методы распознавания»**

**Цель работы:** Изучение методы и алгоритмы распознавания, применяемых в экспертных системах. Изучить методы обучения ЭС на основе алгоритмов распознавания.

## **1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

### **1.1 Назначение методов распознавания**

Системы распознавания предназначены для отнесения распознаваемого объекта или явления к одному из известных классов на основе признаков данного объекта (явления). Под классом здесь понимается некоторая совокупность объектов, обладающих близкими (в чем-либо) свойствами. В качестве объекта может рассматриваться реальный физический объект (например, летательный аппарат) или некоторая ситуация (неисправность, заболевание, данные метеорологических наблюдений и т.д.).

Примеры применения систем распознавания – системы технической диагностики, системы медицинской диагностики, системы технического контроля, системы для составления метеорологических прогнозов и т.д.

Во многих случаях распознаваемый объект можно представить в виде набора чисел – значений признаков этого объекта, т.е. как вектор

$$X = (X_1, X_2, \dots, X_N),$$

где  $X_j$  – значение  $j$ -го признака объекта,  $N$  – количество используемых признаков.

### **1.2 Решающие функции**

Решающая (дискриминаторная, классифицирующая) функция – это функция от значений признаков распознаваемого объекта, по значению которой может быть принято решение об отнесении объекта к одному из известных классов (или о том, что объект не может быть отнесен ни к одному из этих классов). Рассмотрим следующие виды решающих функций:

- решающие функции на основе минимального расстояния;
- разделяющие решающие функции (границы между классами).



### 1.3 Решающие функции на основе минимального расстояния

Решающие функции на основе минимального расстояния могут применяться, если для каждого класса можно указать объект, который может рассматриваться как наиболее характерный представитель (прототип) данного класса.

Пусть распознаваемый объект может относиться к одному из  $M$  классов. Для распознавания используется  $N$  признаков. Пусть для каждого класса известен объект-прототип со значениями признаков  $P_i = (P_{i1}, P_{i2}, \dots, P_{iN})$ . Для распознавания объекта используются следующие функции:

$$D_i = 2 (X_1 \cdot P_{i1} + X_2 \cdot P_{i2} + \dots + X_N \cdot P_{iN}) - (P_{i1} \cdot P_{i1} + P_{i2} \cdot P_{i2} + \dots + P_{iN} \cdot P_{iN}),$$

где  $X_j$  – значения признаков распознаваемого объекта,  $j=1, N$ .

Решающая функция такого вида строится для каждого класса. Таким образом, находится  $M$  таких функций. Объект относится к классу, для которого значение функции  $D_i$  *максимально*. Смысл решающих функций такого вида следующий. Представим распознаваемый объект  $X$  и объекты-прототипы классов  $P_i$  ( $i=1, M$ ) как точки в  $N$ -мерном пространстве (т.е. как точки, имеющие  $N$  координат). Можно доказать, что значение функции  $D_i$  тем *больше*, чем *меньше* расстояние между распознаваемым объектом  $X$  и объектом-прототипом  $P_i$ , вычисляемое по обычной формуле евклидова расстояния:

$$|X - P_i| = \sqrt{(X_1 - P_{i1})^2 + (X_2 - P_{i2})^2 + \dots + (X_n - P_{in})^2}$$

Здесь  $|X - P_i|$  – расстояние между объектами  $X$  и  $P_i$ .

Пример. Разрабатывается экспертная система для проектирования объектов складского хозяйства на промышленных предприятиях. Большинство складов строится по одному из трех типовых проектов (ТП1, ТП2, ТП3). Выбор типового проекта, по которому будет строиться склад, производится на основе анализа двух характеристик склада: грузооборота (тыс. тонн в год) и необходимого запаса единовременного хранения (тонн), т.е. количества груза, которое должно храниться на складе постоянно. Имеются сведения о девяти складах, построенных по трем указанным проектам. Эксплуатация этих складов показала, что проекты были выбраны удачно. Характеристики складов приведены в табл. 1.1. Так как используемые в задаче признаки (грузооборот и запас единовременного хранения) имеют разную размерность и существенно различаются по порядку величин, их необходимо привести к безразмерному виду. Один из способов такого преобразования – деление всех имеющихся величин на максимальное из значений соответствующего признака. В данном примере необходимо разделить все значения грузооборота на 32 тыс. тонн, а значения запаса еди-

новременного хранения – на 800 тонн. В результате будут получены безразмерные величины, находящиеся в диапазоне от нуля до единицы. Результаты приведения к безразмерному виду представлены в табл. 1.2.

Таблица 1.1 – Характеристики складов

Номер склада	Проект	Грузооборот тыс. тонн в год	Запас единовр. хранения, тонн
1	ТП1	4	400
2	ТП1	6	500
3	ТП1	7	300
4	ТП2	14	200
5	ТП2	18	300
6	ТП2	25	350
7	ТП3	23	620
8	ТП3	32	570
9	ТП3	30	800
	max	32	800

Таблица 1.2 – Характеристики складов, приведенные к безразмерному виду

Номер склада	Проект	Грузооборот тыс. тонн в год	Запас единовр. Хранения, тонн
1	ТП1	0,13	0,5
2	ТП1	0,19	0,63
3	ТП1	0,22	0,38
4	ТП2	0,44	0,25
5	ТП2	0,56	0,38
6	ТП2	0,78	0,44
7	ТП3	0,72	0,78
8	ТП3	1,0	0,71
9	ТП3	0,94	1,0
	max	32	800

Информацию об успешно реализованных проектах складов можно использовать в ЭС, применяемой для проектирования новых складов. Такая ЭС будет работать на основе методов распознавания. В качестве классов будут рассматриваться типовые проекты ТП1, ТП2, ТП3, а в качестве распознаваемых объектов – новые (проектируемые) склады. Для проектируемого склада требуется указать планируемые величины грузооборота и запаса единовременного хранения. Задача ЭС будет состоять в том, чтобы на основе анализа этих двух признаков определить, к какому из типовых проектов *ближе* проектируемый склад.

Составим решающие функции для выбора типового проекта на основе минимального расстояния, т.е. на основе минимального различия между проектируемым складом и некоторым типичным складом, построенным по анализируемому типовому проекту. В качестве прототипов (т.е. наиболее типичных складов для каждого из проектов) будем использовать "гипотетические" склады, характеристики которых соответствуют *средним* характеристикам складов, построенных по каждому из проектов. Например, для проекта ТП1 (или для первого класса) объект-прототип будет иметь следующие значения признаков:

$$P_{11} = (0,13+0,19+0,22)/3 = 0,18 \text{ (по признаку } X_1);$$

$$P_{12} = (0,5+0,63+0,38)/3 = 0,50 \text{ (по признаку } X_2).$$

Таким образом,  $P_1 = (0,18; 0,50)$ . Аналогично найдем признаки объектов-прототипов для второго и третьего классов:  $P_2 = (0,59; 0,36)$ ,  $P_3 = (0,89; 0,83)$ .

Составим решающие функции для определения наиболее подходящего проекта на основе минимального расстояния:

$$D_1 = 2 (X_1 \cdot 0,18 + X_2 \cdot 0,5) - (0,18 \cdot 0,18 + 0,5 \cdot 0,5) = 0,36 \cdot X_1 + 1 \cdot X_2 - 0,28;$$

$$D_2 = 2 (X_1 \cdot 0,59 + X_2 \cdot 0,36) - (0,59 \cdot 0,59 + 0,36 \cdot 0,36) = 1,18 \cdot X_1 + 0,72 \cdot X_2 - 0,48;$$

$$D_3 = 2 (X_1 \cdot 0,89 + X_2 \cdot 0,83) - (0,89 \cdot 0,89 + 0,83 \cdot 0,83) = 1,78 \cdot X_1 + 1,66 \cdot X_2 - 1,48.$$

Пусть, например, требуется выбрать наиболее подходящий типовой проект для проектируемого склада. Предполагается, что грузооборот склада будет составлять примерно 12 тыс. тонн в год, а запас единовременного хранения – 200 тонн. Проектируемый склад рассматривается как распознаваемый объект. Требуется определить, к какому классу (типовому проекту) относится этот объект. Перейдем к безразмерным значениям признаков объекта:

$$X_1 = 12/32 = 0,38,$$

$$X_2 = 200/800 = 0,25.$$

Для распознавания воспользуемся решающими функциями:

$$D_1 = 0,36 \cdot 0,38 + 1 \cdot 0,25 - 0,28 = 0,11;$$

$$D_2 = 1,18 \cdot 0,38 + 0,72 \cdot 0,25 - 0,48 = 0,15;$$

$$D_3 = 1,78 \cdot 0,38 + 1,66 \cdot 0,25 - 1,48 = -0,39.$$

Таким образом, распознаваемый объект отнесен ко второму классу. В данном случае это значит, что для проектируемого склада следует выбрать типовой проект ТП2.

## 1.4 Разделяющие решающие функции

Под разделяющими решающими функциями будем понимать решающие функции, представляющие собой уравнения границ, разделяющих классы. Если распознавание объектов выполняется на основе двух признаков (т.е. распознаваемые объекты могут рассматриваться как точки на плоскости), то решающая функция представляет собой уравнение линии на плоскости (в простейшем случае – уравнение прямой). Если используются три признака, то решающая функция – это уравнение некоторой поверхности в трехмерном пространстве и т.д. Методы построения таких функций лучше всего разработаны для случаев, когда имеются только два класса. В таких случаях строится одна решающая функция, принимающая положительные значения при подстановке в нее значений признаков объектов из одного класса, и отрицательные – для другого класса.

### 1.4.1 Методы построение разделяющих решающих функций

Большинство методов построения разделяющих решающих функций основано на использовании *обучающего множества*, т.е. набора объектов, для которых уже *известно*, к каким классам они относятся. В обучающем множестве должны быть объекты из всех имеющихся классов. Построение решающей функции состоит в подборе такого вида функции, при котором она правильно распознает объекты из обучающего множества.

Общий принцип построения разделяющих решающих функций (для двух классов) следующий. Один из классов (любой) обозначается как первый; для объектов из этого класса решающая функция должна принимать положительные значения, для объектов из другого класса – отрицательные. Выбирается некоторый первоначальный вид решающей функции. В нее подставляются объекты из обучающего множества. Если решающая функция правильно распознает объект (т.е. принимает положительное значение для объекта из первого класса, или отрицательное – для объекта из второго класса), то она не изменяется. В случае неправильного распознавания объекта из первого класса (т.е. при *отрицательном или нулевом* значении функции для такого объекта) функция корректируется таким образом, чтобы ее значение *увеличилось*. В случае ошибки для объекта из второго класса (т.е. при *положительном или нулевом* значении функции для такого объекта) функция корректируется таким образом, чтобы ее значение *уменьшилось*. Процесс завершается, когда решающая функция правильно распознает *все* объекты из обучающего множества.

### 1.4.2 Алгоритм восприятия

Рассмотрим один из алгоритмов построения решающих функций – алгоритм восприятия. Воспользуемся им для построения линейной решающей функции между двумя классами (первым и вторым):

$$D_{12} = W_1 \cdot X_1 + W_2 \cdot X_2 + \dots + W_N \cdot X_N + W_0,$$

где  $X_1, X_2, \dots, X_N$  – признаки распознаваемого объекта;

$W_0, W_1, W_2, \dots, W_N$  – искомые коэффициенты решающей функции.

Алгоритм реализуется в следующем порядке.

1) Выбираются некоторые начальные значения коэффициентов решающей функции. Пусть, например, эти коэффициенты равны нулю:

$$W_0 = W_1 = W_2 = \dots = W_N = 0.$$

2) Счетчик правильно распознанных объектов принимается равным нулю:  $E=0$ .

3) Из обучающего множества выбирается первый объект.

4) Выбранный объект подставляется в решающую функцию. В зависимости от значения решающей функции выполняются следующие действия.

– Если объект распознан правильно (т.е.  $D_{12} > 0$  для объекта первого класса, или  $D_{12} < 0$  для объекта второго класса), то решающая функция не изменяется. Счетчик правильно распознанных объектов увеличивается на единицу ( $E=E+1$ ).

– Если неправильно распознан объект из первого класса (т.е. для такого объекта  $D_{12} \leq 0$ ), то функция корректируется в направлении увеличения. Для этого к ее коэффициентам  $W_1, W_2, \dots, W_N$  прибавляются значения признаков распознаваемого объекта  $X_1, X_2, \dots, X_N$ , а коэффициент  $W_0$  увеличивается на единицу. Счетчик правильно распознанных объектов сбрасывается в ноль ( $E=0$ ).

– Если неправильно распознан объект из второго класса (т.е. для такого объекта  $D_{12} \geq 0$ ), то функция корректируется в направлении уменьшения. Для этого из коэффициентов  $W_1, W_2, \dots, W_N$  вычитаются значения признаков распознаваемого объекта  $X_1, X_2, \dots, X_N$ , а коэффициент  $W_0$  уменьшается на единицу. Счетчик правильно распознанных объектов сбрасывается в ноль ( $E=0$ ).

5. Если счетчик правильно распознанных объектов равен общему количеству объектов в обучающем множестве, то алгоритм завершается. Решающая функция построена.

6. Если счетчик правильно распознанных объектов меньше общего количества объектов в обучающем множестве, то выбирается очередной объект (если все объекты в обучающем множестве уже рассмотрены, то снова выбирается первый объект). Выполняется возврат на шаг 4. Таким образом, алгоритм завершается, когда решающая функция правильно распознает все объекты в обучающем множестве. Другими словами, построение решающей функции закан-

чивается после того, как в нее подставляются все объекты обучающего множества, и ни разу не требуется ее корректировка.

Рассмотрим построение решающей функции для первого и второго классов ( типовые проекты ТП1 и ТП2) в приведенном выше примере. Значения коэффициентов решающей функции считаем сначала равными нулю:

$$D_{12}=0 \cdot X_1+0 \cdot X_2+0.$$

Выбираем первый объект из обучающего множества (0,13; 0,5) и подставляем его признаки в  $D_{12}$ . Очевидно, что решающая функция равна нулю. Так как рассматривался объект из первого класса, и для него решающая функция должна быть положительной, выполняем коррекцию решающей функции, как показано выше (шаг 4 алгоритма). Функция принимает вид:  $D_{12}=0,13 \cdot X_1+0,5 \cdot X_2+1$ . Так как проверенный объект был распознан неправильно, счетчик правильно распознанных объектов равен нулю:  $E=0$ .

Выбираем очередной объект:  $X=(0,19; 0,63)$ . Подставляем его в решающую функцию:  $D_{12}=0,13 \cdot 0,19+0,5 \cdot 0,63+1 = 1,34$ . Таким образом,  $D_{12}>0$  для объекта из первого класса. Значит, объект распознан правильно и коррекция функции не требуется. Счетчик увеличивается на единицу:  $E=1$ .

Выбираем очередной объект:  $X=(0,22; 0,38)$ . Подставляем его в решающую функцию:  $D_{12}=0,13 \cdot 0,22+0,5 \cdot 0,38+1 = 1,22 > 0$ . Объект распознан правильно, коррекция функции не требуется. Счетчик увеличивается на единицу:  $E=2$ .

Выбираем очередной объект:  $X=(0,44; 0,25)$ . Подставляем его в решающую функцию:

$$D_{12}=0,13 \cdot 0,44+0,5 \cdot 0,25+1 = 1,18 > 0.$$

Объект распознан неправильно: решающая функция должна быть отрицательной, так как объект относится ко второму классу. Выполняется коррекция функции, как показано в алгоритме (шаг 4). Функция принимает вид:  $D_{12}= -0,31 \cdot X_1+0,25 \cdot X_2$ . Счетчик принимается равным нулю ( $E=0$ ). Это означает, что решающая функция изменилась, и ее требуется проверять для всех объектов заново.

Выбираем очередной объект:  $X=(0,56; 0,38)$ . Для него

$$D_{12}=-0,31 \cdot 0,56 + 0,25 \cdot 0,38 = -0,08 < 0.$$

Объект распознан правильно, коррекция не требуется,  $E=1$ , и т.д.

Таким образом, решающая функция проверена на всех объектах обучающего множества, и все объекты оказались распознанными правильно. Решающая функция имеет следующий вид:

$$D_{12} = -0,31 \cdot X_1 + 0,25 \cdot X_2.$$

Если имеются только два класса, то построенная решающая функция может применяться для распознавания объектов. Распознавание выполняется подстановкой значений признаков объекта в функцию. Если значение функции оказывается положительным, то объект относится к первому классу, если отрицательным – ко второму.

*Геометрическая интерпретация* данного метода распознавания следующая. Пусть объекты рассматриваются как точки в  $N$ -мерном пространстве, где  $N$  – количество признаков объектов. Решающая функция представляет собой границу между классами; это значит, что объекты одного класса находятся «с одной стороны» от решающей функции, а объекты другого класса – с другой стороны. При этом функция построена таким образом, что для объектов первого класса она принимает положительные значения, а для второго – отрицательные. Таким образом, если при подстановке признаков некоторого объекта функция принимает положительное значение, это значит, что объект находится «с той же стороны» от решающей функции, что и объекты первого класса, входившие в обучающее множество.

Воспользуемся построенной решающей функцией, чтобы выбрать типовой проект для склада с грузооборотом 12 тыс. тонн в год и запасом единовременного хранения 200 тонн. Безразмерные значения признаков объекта:  $X_1 = 0,38$ ;  $X_2 = 0,25$  (см. пример в подразделе 1.3). Найдем значение решающей функции:  $D_{12} = -0,31 \cdot 0,38 + 0,25 \cdot 0,25 = -0,06$ . Таким образом, если бы использовались только два типовых проекта, то для склада был бы выбран проект ТП2. Однако в данной задаче делать вывод о выборе типового проекта пока нельзя, так как при построении решающей функции не учитывался типовой проект ТП3 (третий класс). В то же время уже можно утверждать, что для склада не будет использован типовой проект ТП1.

Заметим, что построение линейных решающих функций возможно не всегда. В некоторых случаях значения признаков объектов таковы, что классы оказываются линейно-неразделимыми. В этих случаях применяются нелинейные решающие функции.

### **1.4.3 Построение разделяющих решающих функций для нескольких классов**

Имеется несколько методов построения таких решающих функций. Один из них состоит в том, что строятся разделяющие функции для каждой пары классов, как показано выше. В рассматриваемом примере эти функции будут иметь следующий вид:

$$\begin{aligned} D_{12} &= -0,31 \cdot X_1 + 0,25 \cdot X_2; \\ D_{13} &= -1,64 \cdot X_1 - 0,34 \cdot X_2 + 1; \\ D_{23} &= -0,4 \cdot X_1 - 1,34 \cdot X_2 + 1. \end{aligned}$$

Распознавание объекта выполняется на основе подстановки признаков объекта во все решающие функции. Объект относится к классу с номером  $k$ , если выполняется условие:

$$\begin{aligned} D_{ki} &> 0, \\ \text{где } i &\text{ – номер класса, } i=1, \dots, M, i \neq k, \end{aligned} \quad (1.1)$$

Геометрическая интерпретация данного способа распознавания следующая: объект относится к классу  $k$ , если он находится «со стороны этого класса» относительно всех границ класса (т.е. всех функций  $D_{ki}$ ).

Воспользуемся построенными решающими функциями для рассматриваемого примера. Найдем значения решающих функций для склада с грузооборотом 12 тыс. тонн в год и запасом единовременного хранения 200 тонн (безразмерные значения признаков  $X_1=0,38$ ,  $X_2=0,25$ ):

$$\begin{aligned} D_{12} &= -0,06; \\ D_{13} &= 0,29; \\ D_{23} &= 0,51. \end{aligned}$$

Таким образом, для склада выбирается типовой проект ТП2, так как выполняются условия  $D_{21} > 0$  (то же самое, что  $D_{12} < 0$ ) и  $D_{23} > 0$ .

Если условие (1.1) не выполняется ни для одного из классов, то распознавание объекта невозможно. Это может означать, что распознаваемый объект сильно отличается от всех классов или, наоборот, близок сразу к нескольким классам. Пусть, например, для некоторого склада решающие функции приняли следующие значения:  $D_{12}=0,17$ ;  $D_{13}=-0,08$ ;  $D_{23}=0,14$ . Для такого склада выбор проекта по построенным решающим функциям невозможен: склад не может быть отнесен ни к первому проекту (т.к.  $D_{13} < 0$ ), ни ко второму (т.к.  $D_{21} < 0$ ), ни к третьему (т.к.  $D_{32} < 0$ ). В этом случае следует использовать другие методы распознавания (например, на основе минимального расстояния).



## **2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

### **2.1 Практическое занятие**

1. Изучить основные особенности задач распознавания.
2. Создать примеры объектов (не менее 10), относящихся к двум классам (обучающее множество), а также значения признаков объекта (не менее трех), подлежащего распознаванию.
3. Составить решающие функции на основе минимального расстояния. Выполнить распознавание заданного объекта.
4. Составить разделяющие решающие функции на основе алгоритма восприятия. Выполнить распознавание заданного объекта.
5. Выполнить геометрическую интерпретацию результатов для двух признаков.

### **2.2 Лабораторная работа**

1. Изучить основные особенности задач распознавания.
2. Создать примеры объектов (не менее 10), относящихся к двум классам (обучающее множество), а также значения признаков объекта (не менее пяти), подлежащего распознаванию.
3. Разработать программу, реализующую алгоритм распознавания для решающей функции на основе минимального расстояния. Обучить программу при помощи объектов из обучающего множества. Выполнить распознавание заданных объектов.
4. Разработать программу, реализующую алгоритм распознавания для решающей функции на основе алгоритма восприятия. Обучить программу при помощи объектов из обучающего множества. Выполнить распознавание заданных объектов.
5. Оценить вычислительную сложность алгоритмов.

## **3. КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Понятие экспертной системы.
2. Принятие решений при помощи экспертной системы.
3. Понятие решающей функции.
4. Решающие функции на основе минимального расстояния.
5. Разделяющие решающие функции.
6. Алгоритм восприятия.
7. Обучение экспертной системы.
8. Построение разделяющих решающих функций для двух классов.
9. Построение разделяющих решающих функций для нескольких классов.
10. Геометрическая интерпретация распознавания на основе алгоритма восприятия.

**Лабораторная работа №8 + Практическое занятие 3**  
**(8 часов + 4 часа)**  
**«Методы кластерного анализа»**

**Цель работы:** Изучить методы и алгоритмы разделения объектов на кластеры.

## **1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

### **1.1 Назначение методов кластерного анализа**

Методы кластерного анализа предназначены для разделения множества анализируемых объектов и явлений (предприятия, характеризующиеся набором показателей; автомобили, имеющие определенные характеристики; болезни, имеющие определенные симптомы; погодные явления, характеризующиеся некоторыми признаками и т.п.) на кластеры, т.е. на группы объектов, схожих друг с другом по каким-либо признакам.

Области применения методов кластерного анализа:

- классификация объектов и явлений в некоторой предметной области;
- упорядочение информации о предметной области;
- сжатие информации о предметной области;
- построение баз знаний.

Постановка задачи кластерного анализа следующая. Имеется  $N$  объектов:  $X_1, X_2, \dots, X_N$ . Для описания объектов используется  $M$  признаков. Каждый объект описывается набором значений признаков:

$$X_j = (X_{1j}, X_{2j}, \dots, X_{Mj}), j=1, N.$$

Требуется разделить объекты на кластеры таким образом, чтобы в каждый кластер входили объекты со сходными значениями признаков.

### **1.2 Метод K средних**

Метод предназначен для разделения объектов на заданное число кластеров. Принцип работы метода следующий. На основе имеющейся информации о предметной области задается количество кластеров ( $K$ ). При этом указывается также содержательный смысл каждого кластера (например, объекты с высоким значением некоторого признака, объекты со средним значением некоторых признаков и т.д.). Для каждого кластера выбирается объект-*прототип* (представитель), т.е. объект, наиболее подходящий для данного класса по значениям признаков. Находится первоначальный вариант разделения объектов на кластеры: каждый объект относится к кластеру, представляемому *ближайшим* объектом-

прототипом. Затем в каждом кластере находится новый прототип со средними (для данного кластера) значениями признаков. Снова выполняется отнесение каждого объекта к кластеру, представляемому ближайшим прототипом.

Процедура повторяется до получения окончательного разбиения, т.е. до тех пор, пока на двух последовательных итерациях метода будет получено одинаковое разбиение.

*Пошаговый алгоритм* разбиения объектов на заданное число кластеров на основе метода  $K$  средних.

1. Номер итерации алгоритма принимается равным нулю:  $s=0$ .
2. Задается количество кластеров ( $K$ ). Для каждого кластера выбирается первоначальный объект-прототип:  $P_k^0$ ,  $k = \overline{1..K}$ .
3. Выполняется переход к очередной итерации алгоритма:  $s=s+1$ .
4. Находятся расстояния от каждого из анализируемых объектов до каждого из объектов-прототипов. Выполняется отнесение каждого объекта к ближайшему кластеру, т.е. к кластеру, для которого расстояние между этим объектом и прототипом кластера минимально.
5. В каждом кластере определяется новый объект-прототип:  $P_k^s$ ,  $k = \overline{1..K}$ . Значение каждого признака этого объекта-прототипа определяется как среднее арифметическое значений этого признака для всех объектов, входящих на текущей итерации в данный кластер.
6. Если объекты-прототипы всех кластеров на данной и на предыдущей итерации совпадают (т.е. выполняется условие  $P_k^s = P_k^{s-1}$ ,  $k = \overline{1..K}$ ), то алгоритм завершается. Если на данной итерации получено разбиение объектов, отличное от предыдущего, то выполняется возврат к шагу 3.

### 1.3 Метод максимина

Метод предназначен для деления объектов на кластеры, причем количество кластеров заранее неизвестно; оно определяется автоматически в процессе разбиения объектов. Принцип работы метода следующий. Выбирается один из объектов (любой); он становится прототипом первого кластера. Находится объект, наиболее удаленный от выбранного; он становится прототипом второго кластера. Все объекты распределяются по двум кластерам; каждый объект относится к кластеру, представленному ближайшим прототипом. Затем в каждом из кластеров находится объект, *наиболее удаленный* от своего прототипа. Если расстояние между этим объектом и прототипом кластера оказывается значительным (превышающим некоторую предельную величину), то объект становится новым прототипом, т.е. образуется новый кластер. После этого распределение объектов по кластерам выполняется заново. Процесс продолжается, пока не будет получено такое разбиение на кластеры, при котором расстояние

от каждого объекта до прототипа кластера не будет превышать заданную предельную величину.

*Пошаговый алгоритм реализации метода максимина.*

1. Выбирается любой из объектов, например, первый в списке объектов ( $X_1$ ). Он становится прототипом первого кластера:  $P_1=X_1$ . Количество кластеров принимается равным единице:  $K=1$ .

2. Определяются расстояния от объекта  $P_1$  до всех остальных объектов:  $D(P_1, X_j)$ ,  $j=1, N$ . Определяется объект, наиболее удаленный от  $P_1$ , т.е. объект  $X_f$ , для которого выполняется условие:  $D(P_1, X_f) = \max D(P_1, X_j)$ . Этот объект становится прототипом второго кластера:  $P_2=X_f$ . Количество кластеров принимается равным двум:  $K=2$ .

3. Определяется пороговое расстояние. Оно принимается равным *половине* расстояния между прототипами  $P_1$  и  $P_2$ :  $T= D(P_1, P_2)/2$ . Эта величина будет использоваться для проверки условия окончания алгоритма.

4. Находятся расстояния от каждого из анализируемых объектов до каждого из имеющихся объектов-прототипов. Выполняется отнесение каждого объекта к ближайшему кластеру, т.е. кластеру, для которого расстояние между этим объектом и прототипом кластера минимально.

5. В каждом кластере определяется объект, наиболее удаленный от прототипа своего кластера. Обозначим эти объекты как  $Y_k$ ,  $k=1, K$  (здесь  $k$  – номер кластера,  $K$  – количество кластеров).

6. Для каждого из наиболее удаленных объектов, найденных на шаге 5, проверяется условие:  $D(P_k, Y_k) < T$ . Если это условие выполняется *для всех кластеров*, то алгоритм *завершается*. Если для некоторого объекта  $Y_k$  это условие не выполняется, то он становится прототипом нового кластера, и количество кластеров увеличивается на единицу ( $K=K+1$ ). В результате этого шага количество кластеров  $K$  увеличивается на число, равное количеству новых кластеров.

7. Находится новое пороговое расстояние. Оно определяется как половина среднего арифметического всех расстояний между прототипами:

$$T = \frac{\sum_{i=1}^{K-1} \sum_{j=i+1}^K D(P_i, P_j)}{K(K-1)}$$

8. Выполняется возврат к шагу 4.

Таким образом, окончательным является разбиение, для которого во всех кластерах расстояние от прототипа кластера до каждого из объектов, входящих в этот кластер (даже до самого удаленного), не превышает некоторой предельной величины (порогового расстояния).

## **2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

### **2.1 Практическое занятие**

1. Изучить основные особенности задач кластерного анализа.
2. Создать примеры объектов (не менее 10), относящихся к нескольким классам (обучающее множество), а также значения признаков объекта (не менее двух), подлежащего распознаванию.
3. Разделить объекты на кластеры методом  $K$  средних.
4. Разделить объекты на кластеры методом максимина.

### **2.2 Лабораторная работа**

1. Изучить основные особенности задач кластерного анализа.
2. Создать примеры объектов (не менее 10), относящихся к двум классам (обучающее множество), а также значения признаков объекта (не менее трех), подлежащего распознаванию.
3. Разработать программу, реализующую алгоритм кластеризации методом  $K$  средних.
4. Разработать программу, реализующую алгоритм кластеризации методом максимина.
5. Оценить вычислительную сложность алгоритмов.

# Лабораторная работа №9 + Практическое занятие 4 (8 часов + 4 часа) «Нейронная сеть Хопфилда»

## 1. Цель работы

Изучение топологии, алгоритма функционирования сети Хопфилда.

## 2. Теоретические сведения

Сеть Хопфилда – однослойная, симметричная, нелинейная, автоассоциативная нейронная сеть, которая запоминает бинарные / биполярные образы. Сеть характеризуется наличием обратных связей. Топология сети Хопфилда показана на рис. 4.1. Информация с выхода каждого нейрона поступает на вход всех остальных нейронов. Образы для данной модификации сети Хопфилда кодируются биполярным вектором, состоящим из 1 и -1.

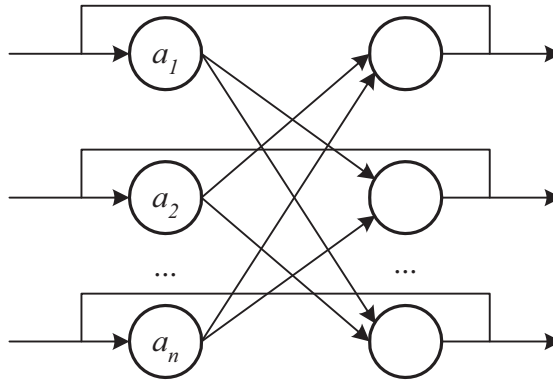


Рис. 4.1. Топология сети Хопфилда

**Обучение.** Обучение сети осуществляется в соответствии с соотношением

$$w_{ij} = \begin{cases} \sum_{k=1}^m a_i^k a_j^k, & i \neq j, \text{ для } i, j = \overline{1, n}, \\ 0, & i = j \end{cases} \quad (4.1)$$

где  $w_{ij}$  – вес связи от  $i$ -го нейрона к  $j$ -му;

$n$  – количество нейронов в сети;

$m$  – количество образов, используемых для обучения сети;

$a_i^k$  –  $i$ -й элемент  $k$ -го образа из обучающей выборки.

Матрица весовых коэффициентов

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & w_{nn} \end{bmatrix}. \quad (4.2)$$

В качестве матрицы весовых коэффициентов Хопфилд использовал симметричную матрицу ( $w_{ij}=w_{ji}$ ) с нулевой главной диагональю ( $w_{ii}=0$ ). Последнее условие соответствует отсутствию обратной связи нейронного элемента на себя. В качестве функции активации нейронных элементов может использоваться как пороговая, так и непрерывная функция, например сигмоидная или гиперболический тангенс.

Будем рассматривать нейронную сеть Хопфилда с дискретным временем. Тогда при использовании пороговой функции активации она называется нейронной сетью с дискретным состоянием и временем. Нейронная сеть с непрерывной функцией активации называется нейронной сетью с непрерывным состоянием и дискретным временем. При использовании непрерывного времени модель Хопфилда называется непрерывной.

Для описания функционирования таких сетей Хопфилд использовал аппарат статистической физики. При этом каждый нейрон имеет два состояния активности ( $1, -1$ ), которые аналогичны значениям спина некоторой частицы. Весовой коэффициент  $w_{ji}$  можно интерпретировать как вклад поля  $j$  – частицы в величину потенциала  $i$  – частицы. Хопфилд показал, что поведение такой сети аналогично поведению лизингового спинового стекла. При этом он ввел понятие вычислительной энергии, которую можно интерпретировать в виде ландшафта с долинами и впадинами. Структура соединений сети определяет очертания ландшафта. Нейронная сеть выполняет вычисления, следуя по пути, уменьшающему вычислительную энергию сети. Это происходит до тех пор, пока путь не приведет на дно впадины. Данный процесс аналогичен скатыванию капли жидкости по склону, когда она минимизирует свою потенциальную энергию в поле тяготения. Впадины и долины в сети Хопфилда соответствуют наборам информации, которую хранит сеть. Если процесс начинается с приближенной или неполной информации, то он следует по пути, который ведет к ближайшей впадине. Это соответствует операции ассоциативного распознавания.

Матрица весов является диагонально симметричной, причем все диагональные элементы равны 0.

**Воспроизведение.** Нейронная сеть Хопфилда может функционировать синхронно и асинхронно. Для воспроизведения используется соотношение

$$a_i(t+1) = f\left(\sum_{j=1}^n w_{ij} a_j(t)\right), \quad (4.3)$$

где  $a_j(t)$  – выход  $j$ -го нейрона в момент времени  $t$ , а  $f$  – бинарная / биполярная функция активации;

$$f(x) = \begin{cases} 1 & x > 0, \\ -1 & x \leq 0. \end{cases} \quad (4.4)$$

При работе в синхронном режиме на один такт работы сети все нейроны одновременно меняют состояние по формуле (4.4). В случае асинхронной работы состояние меняет только один случайный нейрон. Итерации продолжаются до тех пор, пока сеть не придет в стабильное состояние.

Во время воспроизведения исходным вектором  $a(0)$  является некоторый тестовый образ, не совпадающий с образами из обучающей выборки. В процессе функционирования по формуле (4.4) сеть должна прийти в состояние, соответствующее образу из обучающей выборки, наиболее похожему на тестовый.

Максимальное количество образов, которое можно запомнить в матрице  $W$  не превышает

$$m = \frac{n}{2 \ln n + \ln \ln n}, \quad (4.5)$$

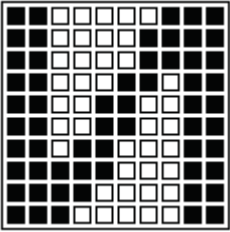
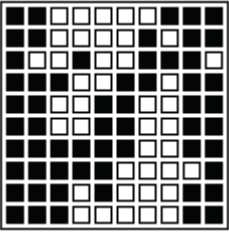
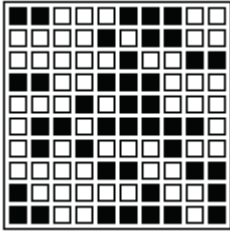
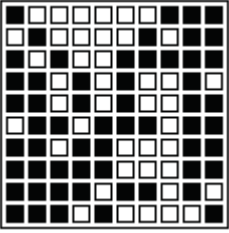
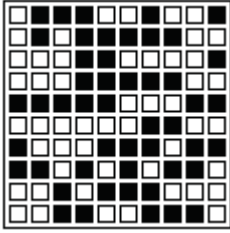
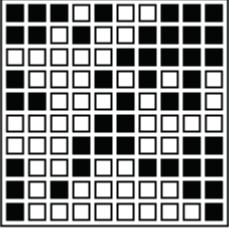
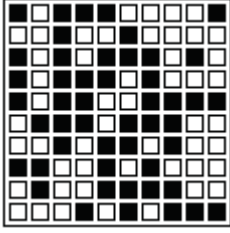
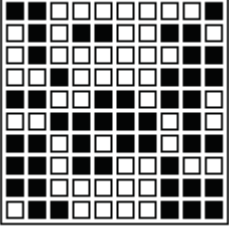
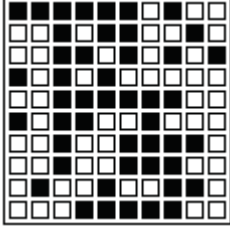
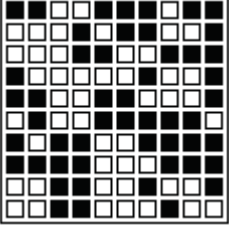
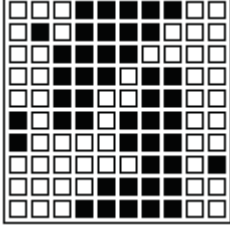
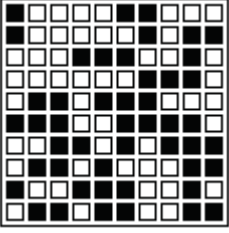
где  $n$  – количество нейронов, что следует отнести к недостаткам этой сети.

### 3. Задание

1. Ознакомьтесь с теоретической частью.
2. На языке программирования напишите программу, реализующую нейронную сеть Хопфилда.
3. Произведите обучение сети Хопфилда на заданный тип образов. Для запоминания в соответствии с вариантом взять 3 образа (первые буквы своей фамилии - бинарные изображения размером 5×7, 10×10 или др.).
4. Подайте на вход сети ряд тестовых образов, в которые внесено зашумление (процент зашумления образа – 10%, 20%, 30%, 35%, 40%, 45%, 50%, 60%, 70%, 80%, 90%, 100%). Тестовых образов должно быть не менее 10 для каждого из классов с одним и тем же процентом зашумления.
5. Проанализируйте результаты, при каком проценте зашумления тестовые образы распознаются верно.



Пример задания тестируемого и искаженных образов

Тестируемый образ	Процент зашумления образа	Вид искаженного образа	Процент зашумления образа	Вид искаженного образа
	10%		50%	
	20%		60%	
	30%		70%	
	35%		80%	
	40%		90%	
	45%		100%	