

Оглавление

Лабораторная работа № 1. Метод полного перебора.....	4
Цель работы.....	4
Теоретические сведения.....	4
Введение.....	4
Задача о шахматном коне.....	4
Задача о восьми ферзях.....	5
Задание.....	5
Контрольные вопросы.....	5
Лабораторная работа № 2. Основы языка Prolog.....	7
Цель работы.....	7
Теоретические сведения.....	7
Введение.....	7
Пример программы: родственные отношения.....	7
Факты.....	7
Вопросы.....	8
Переменные.....	9
Конъюнкция целей.....	9
Правила.....	10
Конъюнкция в правилах.....	11
Переменные в теле правила.....	12
Задание.....	13
Контрольные вопросы.....	13
Лабораторная работа № 3. Решение логических задач на Prolog.....	15
Цель работы.....	15
Теоретические сведения.....	15
Загадка Эйнштейна.....	15
Необходимые элементы Prolog.....	16
Задание.....	17

Контрольные вопросы.....	17
Лабораторная работа № 4. Распознавание образов методом потенциал- ных функций.....	18
Цель работы.....	18
Теоретические сведения.....	18
Задание.....	21
Контрольные вопросы.....	22
Лабораторная работа № 5. Нейронные сети. Обучение персептрона.....	23
Цель работы.....	23
Теоретические сведения.....	23
Задание.....	25
Контрольные вопросы.....	26
Лабораторная работа № 6. Нейронные сети. Обучение без учителя.....	27
Цель работы.....	27
Теоретические сведения.....	27
Задание.....	32
Контрольные вопросы.....	32

Лабораторная работа № 1. Метод полного перебора

Цель работы

Ознакомиться с методом полного перебора

Теоретические сведения

Введение

Существуют задачи, которые называются плохо формализуемыми или даже неформализуемыми. Это задачи, в которых не полностью определены условия, не все связи между исходными данными заданы в аналитическом виде. Для таких задач не существует четких алгоритмов и для их решения применяются методы искусственного интеллекта.

Одним из типов задач, для которых впервые начали применяться методы искусственного интеллекта, были шахматные задачи, а одним из самых первых подходов к их решению являлся метод полного перебора. Его суть заключается в получении решения задачи путем перебора всех возможных вариантов. Ясно, что сложность такого метода зависит от полного числа всех возможных решений.

Рассмотрим кратко две классические задачи, которые можно решить методом полного перебора с отходом назад (поиск с возвратом): задача о шахматном коне и задача о восьми ферзях.

Задача о шахматном коне

Требуется обойти конем всю шахматную доску, не побывав при этом на одном и том же поле более одного раза. Опишем алгоритм решения задачи методом полного перебора с отходом назад.

Пронумеруем все восемь возможных ходов коня. С текущей клетки C_n совершаем первый по номеру возможный ход, т. е. такой ход, при котором конь не

покидает пределы доски и не становится на уже пройденное поле. Пусть номер такого хода равен m . Если с новой клетки C_n+1 есть возможные ходы — то снова совершаем первый по номеру возможный ход и т. д. Если же с новой текущей клетки C_n+1 невозможно совершить ход, не выходя при этом за пределы доски и не попадая на уже пройденное поле, то возвращаемся на ход назад, на клетку C_n , и делаем из нее следующий по номеру возможный ход $m+1$. Очевидно, что наиболее просто реализовать этот алгоритм с помощью рекурсии.

Задача о восьми ферзях

Требуется расставить на шахматной доске восемь ферзей так, чтобы ни один из них не находился под ударом любого другого. Опишем алгоритм решения задачи методом полного перебора с отходом назад. Первый ферзь ставится на первую горизонталь шахматной доски, затем каждый следующий ферзь ставится на следующую горизонталь так, чтобы не оказаться под ударом уже поставленных ферзей. Если на каком-то шаге поставить очередного ферзя нельзя, то происходит отход назад и корректировка положения предыдущего ферзя. Таким образом, имеется сходство с алгоритмом решения задачи о шахматном коне, поскольку оба алгоритма относятся к поиску с возвратом.

Задание

1. Изучить теоретические сведения.
2. По выбору преподавателя реализовать один из описанных алгоритмов в виде компьютерной программы. Предусмотреть возможность визуализации решения и процесса его поиска.
3. Составить отчет о выполнении работы, включающий в себя словесное описание алгоритма и блок-схему программы.
4. Подготовить ответы на контрольные вопросы.

Контрольные вопросы

1. Для каких задач применяются методы искусственного интеллекта?

2. Что такое метод полного перебора?
3. В чем заключается суть поиска с возвратом?
4. Под эвристикой понимают правило, следуя которому можно уменьшить количество вариантов перебора. Для реализованного алгоритма привести хотя бы одну эвристику и реализовать ее в программе.

Лабораторная работа № 2. Основы языка Prolog

Цель работы

Ознакомиться с языком Prolog.

Теоретические сведения

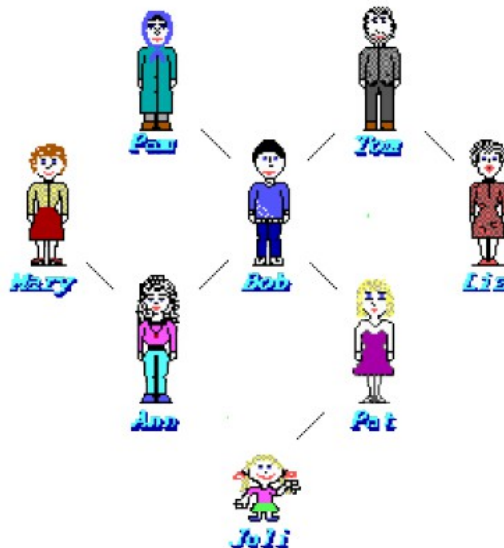
Введение

Prolog — это язык программирования, используемый для решения задач, в которых действуют объекты и отношения между этими объектами.

Программа на Prolog состоит из предложений, которые могут быть фактами, правилами или вопросами.

Пример программы: родственные отношения

Рассмотрим дерево родственных отношений:



Факты

Введем отношение «родитель» (parent) между объектами:

`parent(tom, bob) .`

Это факт, определяющий, что Том является родителем Боба.

`parent` — имя отношения, `tom`, `bob` — его аргументы. Теперь можно записать программу, описывающую все дерево родственных отношений.

```
parent(pam, bob) .  
parent(tom, bob) .  
parent(tom, liz) .  
parent(bob, ann) .  
parent(bob, pat) .  
parent(mary, ann) .  
parent(pat, juli) .
```

Эта программа состоит из семи предложений (утверждений, клауз). Каждая клауза записана фактом в виде отношения `parent`.

При записи фактов надо соблюдать следующие правила:

1. Имена всех отношений и объектов с маленькой буквы.
2. Сначала записывается имя отношения, затем в круглых скобках через запятую объекты.
3. В конце ставится точка.

Еще пример факта:

```
like(bob, pam) .
```

Совокупность фактов в Prolog называют базой знаний.

Вопросы

К составленной базе знаний можно задать вопросы. Вопрос в обычном Prolog начинается с `?` –

Вопрос записывается также, как и факт. Например:

```
? - parent(bob, pat) .  
yes
```

Когда Prolog получает вопрос, он пытается сопоставить его с базой данных. Такой факт находится, ответ: да (`yes`). На вопрос

```
?-parent(bob, mary) .  
no
```

Ответ будет «нет» (no), так как такого факта в базе данных нет.

Переменные

Можно задать вопрос и узнать кто родитель liz:

```
?-parent(X, liz).
```

```
X= tom
```

Здесь X — переменная. Ее величина неизвестна и она может принимать значения. В данном случае ее значением будет объект, для которого это утверждение истинно. Вопрос:

```
?-parent(X, bob).
```

```
X=tom
```

```
X=pam
```

Можно задать вопрос, кто является чьим родителем. Или найти такие X и Y, что X является родителем Y.

```
?-parent(X, Y).
```

```
X= pam
```

```
Y= bob
```

```
Y= tom
```

```
X= bob
```

```
и т. д.
```

Конъюнкция целей

Можно задать более общий вопрос: Кто является родителем родителя juli. Так как нет отношения grandparent, то можно разбить на два вопроса: кто родитель juli. Предположим — Y. кто родитель Y. Предположим — X.

Тогда составной вопрос:

```
?-parent(Y, juli), parent(X, Y).
```

```
X=bob
```

```
Y=pat
```

При поиске решения сначала находится Y, а затем по второму условию X.

Вопрос: Кто внуки тома?:

```
?-parent (tom, Y) , parent (Y, X) .
```

```
Y=bob
```

```
X=ann
```

```
Y=bob
```

```
X=pat
```

И наконец, есть ли у ann и pat общий родитель?

```
?-parent (Y, ann) , parent (Y, pat) .
```

```
Y=bob
```

Правила

Введем отношение «ребенок» `child`, обратное к `parent` «родитель».

Можно было бы определить аналогично:

```
child (liz, tom) .
```

Но можно использовать, что отношение `child` обратно к `parent` и записать в виде утверждения-правила:

```
child(Y, X) :-parent (X, Y) .
```

Правило читается так:

Для всех X и Y

Y — `child` X , если

X — `parent` Y .

Правило отличается от факта тем, что факт всегда истина, а правило описывает утверждение, которое будет истинной, если выполнено некоторое условие. Если условие `parent (X, Y) .` выполняется, то логическим следствием из него будет утверждение `child (Y, X) .`

Как правило используется в Prolog? Зададим вопрос

```
?-child(liz, tom) .
```

В программе нет данных о `child`. Но есть правило, которое верно для всех X и Y , в том числе для `liz` и `tom`. Мы должны применить правило для

этих значений. Для этого надо подставить в правило вместо X — tom, а вместо Y — liz. Говорят, что переменные будут связаны, а операция будет называться подстановкой. Получаем конкретный случай для правила `child(liz, tom) :- parent(tom, liz).`

Условная часть приняла вид `parent (tom, liz)`. Теперь надо выяснить, выполняется ли это условие. Исходная цель `child(liz, tom)` заменяется подцелью `parent (tom, liz)`, которая выполняется, поэтому Prolog ответит «yes».

Конъюнкция в правилах

Добавим еще одно отношение в базу данных, унарное, определяющее пол.

```
male(tom) .  
male(bob) .  
male(jim) .  
female(liz) .  
female(pam) .  
female(pat) .  
female(ann) .
```

Теперь определим отношение `mother`. Оно описывается следующим образом:

Для всех X и Y
 X — mother Y , если
 X — parent Y и
 X — female.

Таким образом правило будет
`mother(X, Y) :- parent(X, Y), female(X) .`

Можно записать
`mother(X, Y) :- parent(X, Y),`

```
female(X) .
```

или

```
mother(X, Y) :-  
parent(X, Y) ,  
female(X) .
```

Запятая между двумя условиями означает конъюнкцию целей. Это означает, что два условия должны быть выполнены одновременно.

Как система ответит на вопрос

```
?-mother(pam, bob) .
```

yes

Находится правило mother, производится подстановка X=pam, Y=bob.

Получаем правило

```
mother(pam, bob) :-  
parent(pam, bob) ,  
female(pam) .
```

Сначала удовлетворяются parent, а затем female. Пролог отвечает: «yes».

Вопрос:

```
?-mother (X, bob) .
```

X=pam

Переменные в теле правила

Определим отношение sister

Для любых X и Y

X sister Y, если

у X и Y есть общий родитель,

и X female

Запишем правило на прологе

```
sister (X, Y) :- parent(Z, X) ,
```

```
parent (Z, Y) ,  
female (X) .
```

Здесь Z — общий родитель. Z — некоторый, любой. Можно спросить

```
?-sister (ann, pat) .
```

```
yes
```

```
?-sister (pat, pat) .
```

```
yes
```

Ответ будет «yes». Так как мы не потребовали, чтобы X и Y были разные.

Добавим отношение

```
different (X, Y) , которое указывает, что X и Y разные.
```

```
sister (X, Y) :- parent (Z, X) ,
```

```
parent (Z, X) ,
```

```
female (X) ,
```

```
different (X, Y) .
```

Задание

1. Изучить теоретические сведения.
2. Создать указанную в теоретических сведениях базу знаний и проверить работоспособность всех приведенных примеров.
3. Составить и протестировать отношения «отец», «дедушка», «бабушка», «брат», «дядя» и «тетя». При необходимости добавить факты в базу знаний.
4. Составить отчет о выполнении работы, включающий в схему родственных отношений (рисунок), код, реализующий указанные выше отношения.
5. Подготовить ответы на контрольные вопросы.

Контрольные вопросы

1. Что такое факт?
2. Что такое правило?
3. Как задать конъюнкцию фактов?

4. Как задать конъюнкцию целей?
5. Как можно использовать переменные в теле правила?

Лабораторная работа № 3. Решение логических задач на Prolog

Цель работы

Научиться применять язык Prolog для решения логических задач на примере «Загадки Эйнштейна».

Теоретические сведения

Загадка Эйнштейна

Загадка Эйнштейна является одной из наиболее распространенных логических головоломок и задач.

Опишем один из вариантов ее условия. На улице стоят подряд пять домов. Все дома разного цвета. В каждом доме живет человек, имеющий уникальную национальность и предпочитающий уникальные марку сигарет и напиток и имеющий уникальное домашнее животное. О домах и их жителях известно следующее:

1. Норвежец живёт в первом доме.
2. Англичанин живёт в красном доме.
3. Зелёный дом находится слева от белого, рядом с ним.
4. Датчанин пьёт чай.
5. Тот, кто курит Marlboro, живёт рядом с тем, кто выращивает кошек.
6. Тот, кто живёт в жёлтом доме, курит Dunhill.
7. Немец курит Rothmans.
8. Тот, кто живёт в центре, пьёт молоко.
9. Сосед того, кто курит Marlboro, пьёт воду.
10. Тот, кто курит Pall Mall, выращивает птиц.
11. Швед выращивает собак.

12. Норвежец живёт рядом с синим домом.
13. Тот, кто выращивает лошадей, живёт в синем доме.
14. Тот, кто курит Winfield, пьет пиво.
15. В зелёном доме пьют кофе.

Один из возможных вопросов этой загадки: кто разводит рыбок? Ясно, что полное решение загадки Эйнштейна можно представить в виде следующей таблицы.

дом	1	2	3	4	5
цвет	желтый	синий	красный	зеленый	белый
национальность	норвежец	датчанин	англичанин	немец	швед
напиток	вода	чай	молоко	кофе	пиво
сигареты	Dunhill	Marlboro	Pall Mall	Rothmans	Winfield
животное	кошка	лошадь	птицы	рыбки	собака

Необходимые элементы Prolog

Для начала нужно заполнить базу знаний о фактах (условиях) задачи. Для этого можно составить правила «первый дом», «дом слева», «дом справа», «соседний дом», описывающие относительное расположение домов.

Очевидным и простым выбором является описание фактов в виде набора (цвет, национальность, напиток, сигареты, животное). Однако по условию задачи мы не можем составить такой набор ни для одного из домов, так как некоторые характеристики не указаны в условии. Для указания того, что значение не известно, используется конструкция `_`, например:

```
house (red, en, _, _, _).
```

Для связанного хранения группы фактов (15 условий задачи) можно использовать списки. Список является объектом, содержащим внутри произвольное число других объектов. Списки соответствуют, грубо говоря, массивам в других языках, но, в отличие от массивов, список не требует декларирования его размера до начала его использования. Список, содержащий числа 1, 2 и 3 записывается как `[1, 2, 3]`.

Если необходимо создать список из пяти неизвестных заранее элементов, можно воспользоваться конструкцией

```
W = [_, _, _, _, _],
```

где *W* – имя списка.

Для добавления элемента *elem* в список *W* используется конструкция *member*:

```
member(elem, W) .
```

Задание

1. Изучить теоретические сведения.
2. Создать на основе условий задачи базу знаний. Для этого рекомендуется использовать рекомендуемый в теоретических сведениях список правил (его может быть недостаточно!)
3. Написать правило для решения загадки Эйнштейна (один из простейших вариантов — с помощью использования списка).
4. Составить отчет о выполнении работы, включающий описание базы знаний и правила для нахождения решения.
5. Подготовить ответы на контрольные вопросы.
6. Для тех, кому показалось мало: реализовать решение загадки Эйнштейна на каком-либо языке программирования, допускающем процедурное программирование, например, на языке С (выполнение этого задания поощряется дополнительными баллами). В отчете представить блок-схему программы.

Контрольные вопросы

1. Каким способом можно задать элемент, значение которого заранее неизвестно?
2. Как создать список?
3. Как добавить элемент в список?

Лабораторная работа № 4. Распознавание образов методом потенциальных функций









Цель работы

Изучение метода потенциальных функций для распознавания образов.

Теоретические сведения

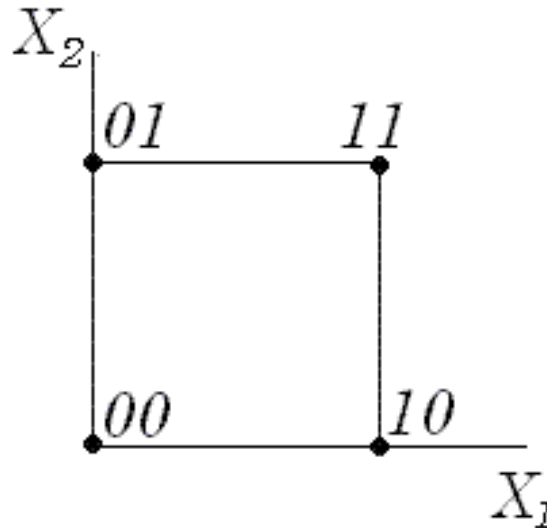
Алгоритм распознавания образов методом потенциальных функций относится к числу так называемых алгоритмов с обучением. Доказано, что он имеет глубокую аналогию с популярной в настоящее время нейросетевой технологией распознавания образов. Рассмотрим его в применении к задаче распознавания графических изображений.

Для простоты мы будем рассматривать только черно-белые изображения. Пусть рисунок состоит всего из двух пикселей. Тогда множество всех объектов, которое можно будет изобразить (универсальное множество), состоит из четырех объектов: $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$, где 1 — черный пиксель, 0 — белый.

X_1	X_2	X_1	X_2
		0	0
		1	0
		0	1
		1	1
А)		Б)	

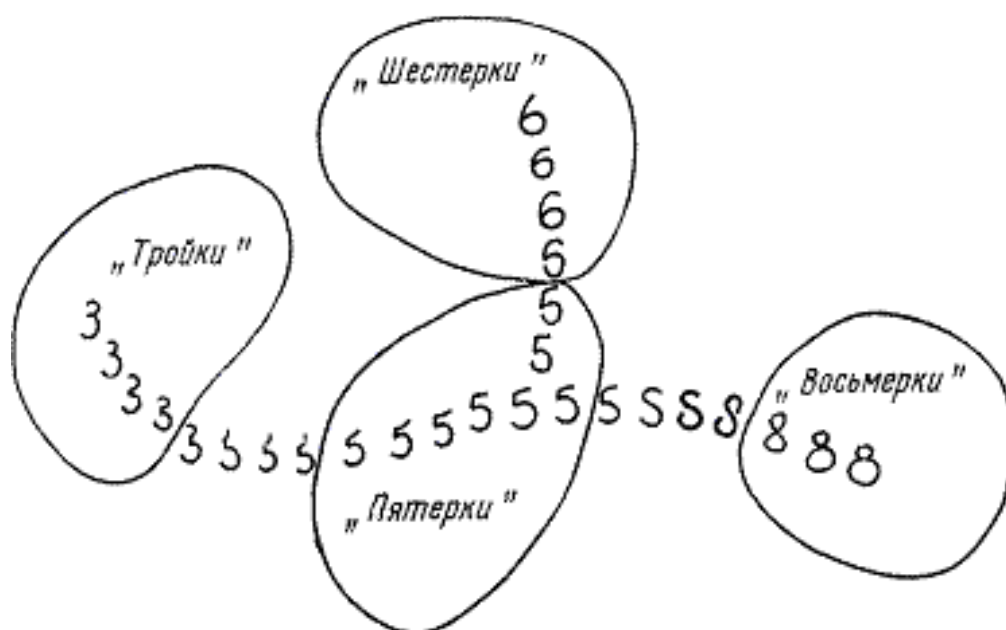
Все объекты универсального множества можно разместить в вершинах единичного квадрата. Таким образом, множеству фигур, изображенных на двух-пиксельном поле, может быть сопоставлено множество точек в двумерном пространстве. Ребру этого квадрата будет соответствовать переход от одного изоб-

ражения к другому. Для перехода от (1,1) к (0,0) нужно будет пройти два ребра, для перехода от (0,1) к (0,0) — одно. Отметим, что число ребер в нашем переходе — это количество несовпадающих пикселей двух изображений. Вывод: расстояние от одного рисунка до другого равно числу несовпадающих пикселей в них. Это расстояние называется расстоянием по Хэммингу.



Теперь представим себе, что у нас рисунок состоит из трех пикселей. Коды изображений тогда будут состоять из трех значений, универсальное множество — из восьми элементов, которые мы разместим в вершинах единичного куба. Но принципиально ничего не изменится, и расстояние по Хэммингу вычисляется так же.

Если в задаче используется рисунок $50 \times 70 = 3500$ пикселей, то в этом случае код любого изображения состоит из 3500 значений, универсальное множество — из 23500 элементов, которые мы будем размещать в вершинах единичного 3500-мерного куба. Основная идея заключается в том, что в этом многомерном кубе изображения, соответствующие какому-то определенному образу, лежат недалеко друг от друга. Эта идея получила название "Гипотеза о компактности образов".



Теперь можно сформулировать задачу: нужно универсальное множество разбить на "куски", компактные множества, каждому из которых соответствует образ.

Реализация алгоритма несложная. Программе в процессе обучения сообщаются изображения (точки многомерного куба) и указания, к какому образу каждое изображение относится. При распознавании программа просто смотрит, в какую из известных компактных областей попало входное изображение. Для этого используется некоторая характеристика, которая показывает удаленность одного рисунка (точки в вершине многомерного куба) до группы таких же изображений. В качестве меры удаленности рисунка от группы рисунков используется потенциал.

Известно, что электрический заряд создает вокруг себя поле, одной из характеристик которого является потенциал. В любой точке он может быть вычислен по формуле

$$P = a \frac{q}{R^2}$$

где a — некоторый постоянный коэффициент, q — величина заряда, R — расстояние от данной точки до заряда. Если электрическое поле образовано двумя или более зарядами, то потенциал в данной точке равен сумме потенциалов

каждого заряда. Аналогия очевидна — каждый рисунок, на котором программа обучалась, создает в пространстве универсального множества потенциал. После обучения программе дают распознать какой-либо рисунок (точку в вершине многомерного куба), программа вычисляет потенциал, создаваемый в этой точке всеми объектами образа "а", образа "б"... на которых программу учили и распознаваемый рисунок относится к образу, который создал наибольший потенциал.

Таким образом, алгоритм представляет собой следующую последовательность операций.

1. Формирование множества распознаваемых образов (например, изображения цифр от 0 до 9) и обучающей выборки для каждого из образов (например, по 10 изображений каждой цифры, нарисованных различными шрифтами).

2. Формирование распознаваемого изображения (например, изображения какой-либо цифры, нарисованной от руки);

3. Вычисление расстояний от распознаваемого изображения до каждого из эталонных изображений обучающей выборки (расстояний по Хэммингу, то есть количества несовпадающих пикселей). Это даст матрицу R_{ij} , где i — номер образа, j — номер обучающего изображения для i -го образа.

4. Вычисление потенциалов, создаваемых обучающими изображениями, в точке распознаваемого изображения по формуле

$$\varphi(R) = \frac{1000000}{1 + R^2}$$

5. Суммирование потенциалов, создаваемых каждым из образов в точке распознаваемого изображения и выбор образа, создающего наибольший потенциал. Это и будет означать, что распознаваемое изображение принадлежит выбранному образу (например, что это — цифра 2).

Задание

Разработать программу, реализующую следующие функции:

1. Определить три образа: круг, треугольник и квадрат

2. Определить обучающую выборку для каждого из образов, состоящую из 5 отличающихся друг от друга изображений соответствующей геометрической фигуры
3. Задать распознаваемое изображение
4. Определить расстояние по Хэммингу от распознаваемого изображения до изображений обучающей выборки
5. Определить потенциал, создаваемый каждым из образов в распознаваемой точке и выбрать образ с наибольшим потенциалом.

Контрольные вопросы

1. К какому типу методов обучения относится метод потенциальных функций?
2. Как определить расстояние по Хэммингу?
3. Опишите основные шаги алгоритма потенциальных функций.
4. Чем вызвано использование столь большой константы в числителе при вычислении потенциала?

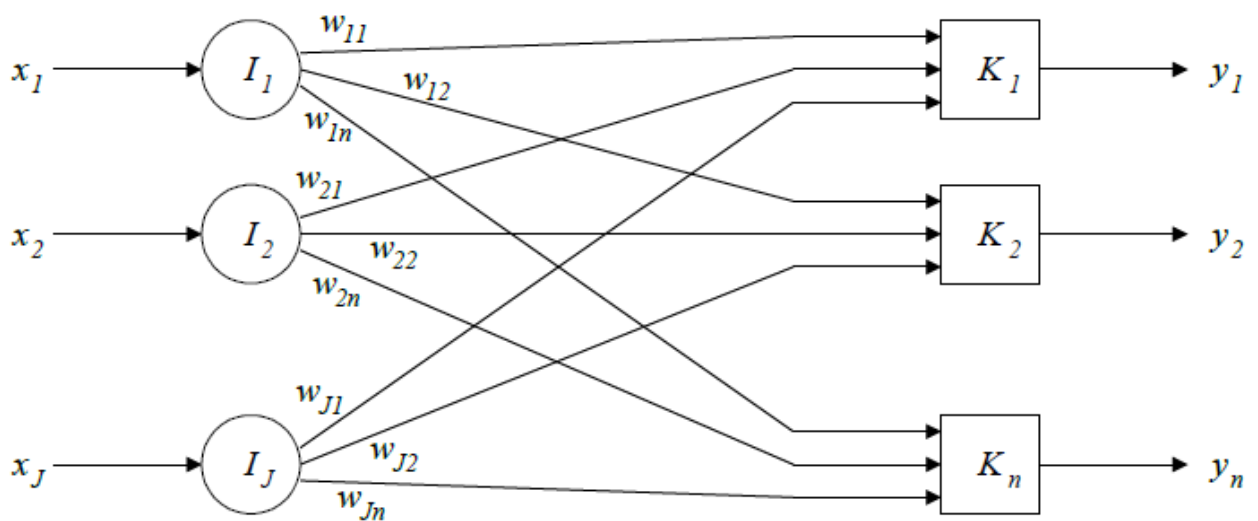
Лабораторная работа № 5. Нейронные сети. Обучение персептрона

Цель работы

Изучение алгоритма обучения персептрона с учителем.

Теоретические сведения

Персептроны являются основой элементной базы современных нейронных сетей. Исторически их появление относят к 1957 году, когда Р. Розенблатт разработал модель, которая вызвала большой интерес у исследователей. Несмотря на некоторые ограничения её исходной формы, она стала основой для многих современных, наиболее сложных алгоритмов обучения с учителем.



Персептрон является двухуровневой нерекуррентной сетью, вид которой показан на рисунке. Она использует алгоритм обучения с учителем. Обучающая выборка состоит из множества входных векторов, для каждого из которых указан свой требуемый вектор цели. Компоненты входного вектора X представлены непрерывным диапазоном значений. Компоненты вектора цели Y являются двоичными величинами (0 или 1). После обучения сеть получает на входе набор непрерывных входов и вырабатывает требуемый выход в виде вектора

с бинарными компонентами.

Обучение осуществляется следующим образом:

1. Все веса сети случайно выбираются как малые величины;
2. На вход сети подаётся входной обучающий вектор X и вычисляется сигнал NET от каждого нейрона, используя стандартное выражение

$$NET_j = \sum_i x_i w_{ij}$$

3. Вычисляется значение пороговой функции активации для сигнала NET от каждого нейрона следующим образом:

$$OUT_j = \begin{cases} 1, & NET_j > \theta_j \\ 0 & \end{cases}$$

Здесь θ_j представляет собой порог, соответствующий нейрону j (в простейшем случае все нейроны имеют один и тот же порог).

4. Вычисляется ошибка для каждого нейрона посредством вычитания полученного выхода из требуемого выхода:

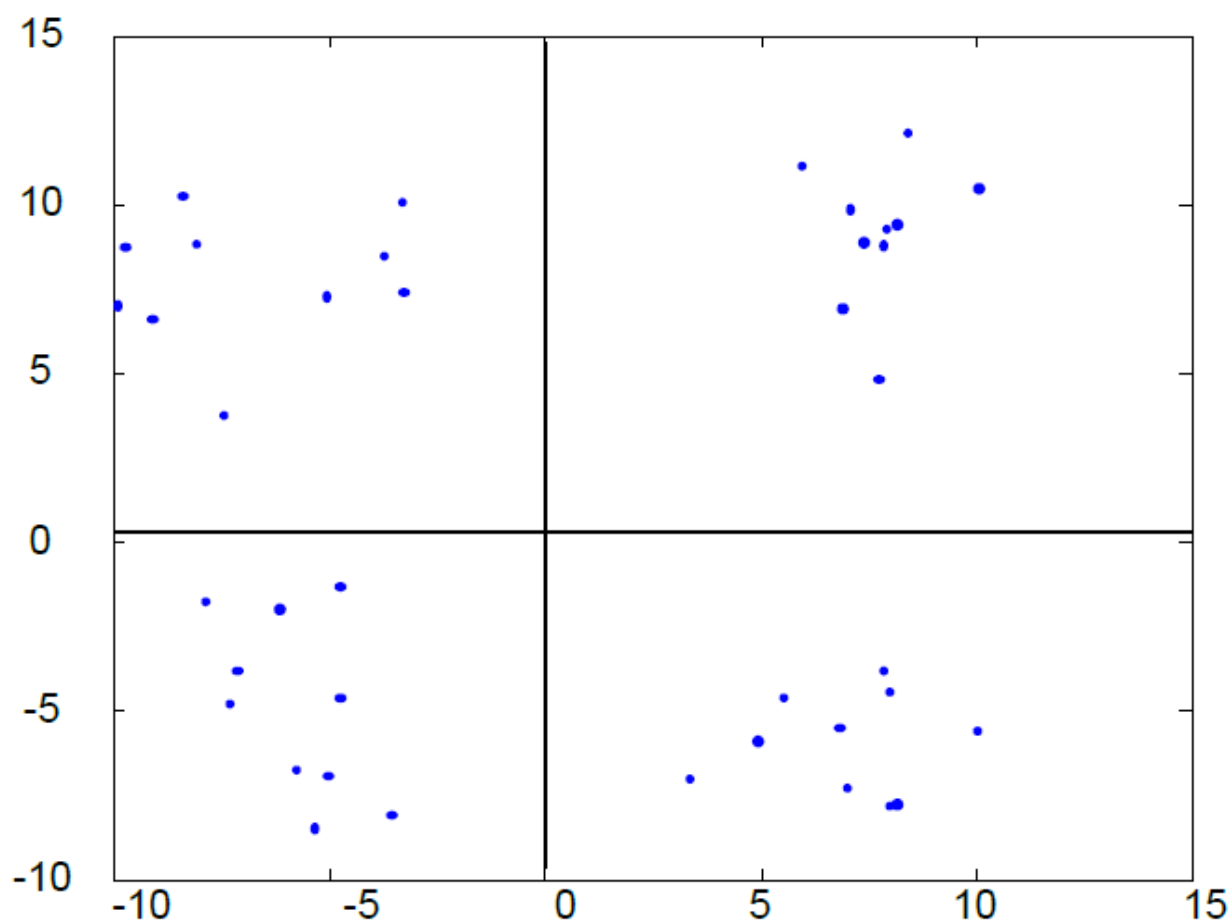
$$error_j = t_j - OUT_j$$

5. Каждый вес модифицируется следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha x_i error_j$$

6. Шаги 2-5 повторяются, пока ошибка не станет достаточно малой.

Рассмотрим обсуждаемый метод для решения задачи классификации. Пусть требуется построить сеть для решения задачи о принадлежности точки к тому или иному квадранту декартовой плоскости (от 1 до 4). Вход сети – координаты x и y точки, выход – вектор из 4-х элементов, причем элемент, соответствующий найденному квадранту точки равен 1, остальные равны 0. Таким образом, сеть будет состоять из 2 входов и 4 выходов. Требуется настроить веса сети так, чтобы она могла корректно распознавать принадлежность точки к тому или иному квадранту.



На рисунке показано расположение 40 случайно выбранных точек – по 10 в каждом квадранте. Это будет обучающей выборкой. Точки «вразнобой» проходят через процедуру обучения 1-5. Коэффициент выбран равным 1. В результате, весовые коэффициенты приобретают следующие значения

$$w_{11} = 11.7035 \quad w_{12} = 5.8339$$

$$w_{21} = -4.7798 \quad w_{22} = -3.4843$$

$$w_{31} = -3.5149 \quad w_{32} = -11.5192$$

$$w_{41} = 8.4805 \quad w_{42} = -10.1324$$

После этого сеть становится пригодной к работе над задачей по распознаванию принадлежности точки к квадранту.

Задание

Задать случайно значения точек в пространстве – по 10 в каждой из восьми областей, образуемых осями координат. Обучить нейронную сеть типа пер-

септрон, состоящую из 3 входов и 8 выходов так, чтобы она могла распознавать в какой из 8 областей пространства находится точка (аналогично рассмотренному примеру для плоскости).

Контрольные вопросы

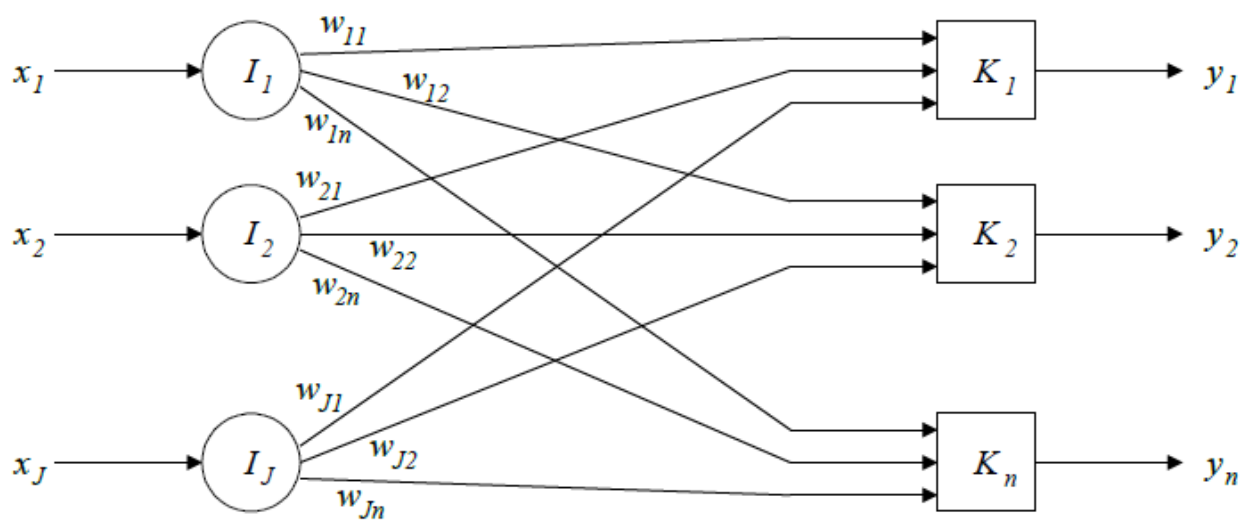
1. Изобразите и опишите структуру нейронной сети.
2. В чем заключается суть алгоритма обучения с учителем?
3. Что такое обучающая выборка и для чего она нужна?

Лабораторная работа № 6. Нейронные сети. Обучение без учителя

Цель работы

Изучение алгоритма обучения нейронной сети Кохонена без использования обучающей выборки.

Теоретические сведения



На рисунке показана структура нейронной сети Кохонена. Входные нейроны I (показанные кружками) служат лишь точками разветвления и не выполняют вычислений. Каждый нейрон этого слоя соединен с каждым нейроном слоя K (называемого слоем Кохонена, сами нейроны K называются нейронами Кохонена) весом w_{ij} . Эти веса в целом рассматриваются как матрица весов W . Сеть Кохонена функционирует в двух режимах: в нормальном, при котором принимается входной вектор X и выдается выходной вектор Y , и в режиме обучения, при котором подается входной вектор и веса корректируются, чтобы дать требуемый выходной вектор. Это напоминает другие разновидности нейронных сетей, различие состоит в операциях, выполняемых нейронами Кохонена.

В своей простейшей форме слой Кохонена функционирует в духе «побе-

дитель забирает все», то есть для данного входного вектора один и только один нейрон Кохонена выдает на выходе логическую единицу, а все остальные выдают ноль. Нейроны Кохонена можно воспринимать как набор электрических лампочек, и для любого входного вектора «загорается» одна из них.

Подобно нейронам большинства сетей, выход NET каждого нейрона Кохонена является просто суммой взвешенных входов. Это может быть выражено следующим образом:

$$NET_j = \sum_i x_i w_{ij}$$

где NET_j - это выход NET нейрона Кохонена с номером j . Нейрон Кохонена с максимальным значением NET_j является «победителем». Его выход равен единице, у остальных он равен 0.

Слой Кохонена классифицирует входные векторы в группы схожих. Это достигается с помощью такой подстройки весов слоя Кохонена, что близкие входные векторы активируют один и тот же нейрон данного слоя. Обучение Кохонена является самообучением, протекающим без учителя. Поэтому не нужно предсказывать, какой именно нейрон Кохонена будет активироваться для заданного входного вектора. Необходимо лишь гарантированно добиться, чтобы в результате обучения разделялись несхожие входные векторы.

Перед тем, как предъявить входные векторы сети, их желательно предварительно обработать, нормализовав. Операция выполняется с помощью деления каждой компоненты входного вектора на длину вектора. Таким образом, входной вектор превращается в единичный вектор с тем же самым направлением, т.е. вектор единичной длины в J -мерном пространстве.

При обучении слоя Кохонена на вход подается входной вектор и вычисляются его скалярные произведения с векторами весов, связанными со всеми нейронами Кохонена. Нейрон с максимальным значением скалярного произведения объявляется «победителем» и его веса подстраиваются. Так как скалярное произведение, используемое для вычисления величин NET , является мерой

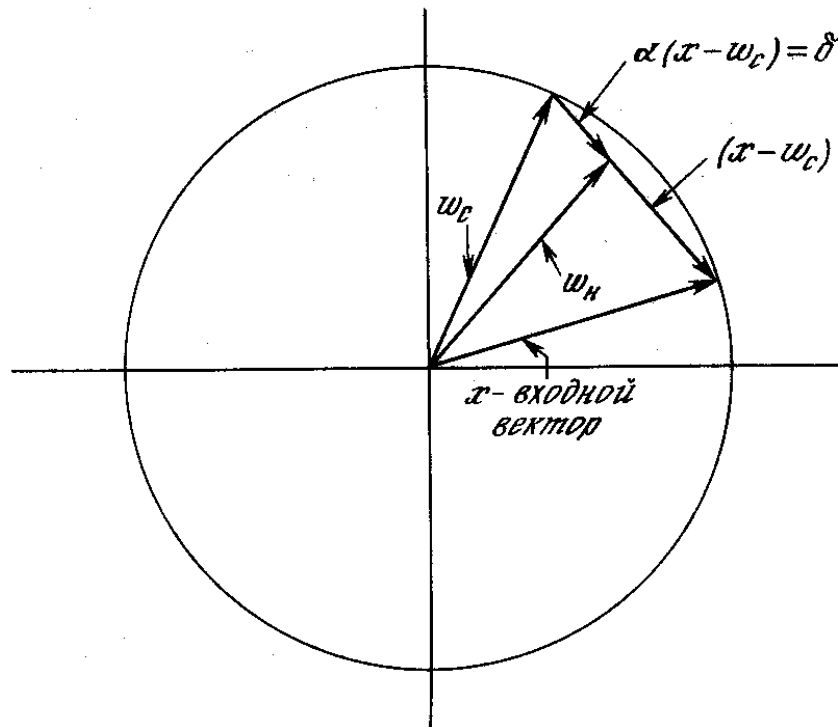
сходства между входным вектором и вектором весов, то процесс обучения состоит в выборе нейрона Кохонена с весовым вектором, наиболее близким к входному вектору, и дальнейшем приближении весового вектора к входному. Снова отметим, что процесс является самообучением, выполняемым без учителя. Сеть самоорганизуется таким образом, что данный нейрон Кохонена имеет максимальный выход для данного входного вектора. Уравнение, описывающее процесс обучения имеет следующий вид:

$$w_n = w_c + \alpha(x - w_c),$$

w_n — новое значение веса, соединяющего входную компоненту x с выигравшим нейроном; w_c — предыдущее значение этого веса; α — коэффициент скорости обучения, который может варьироваться в процессе обучения.

Каждый вес, связанный с выигравшим нейроном Кохонена, изменяется пропорционально разности между его величиной и величиной входа, к которому он присоединен. Направление изменения минимизирует разность между весом и его входом.

На рисунке этот процесс показан геометрически в двумерном виде. Сначала находится вектор $X - W_c$, для этого проводится отрезок из конца W в конец X . Затем этот вектор укорачивается умножением его на скалярную величину α , меньшую единицы, в результате чего получается вектор изменения δ . Окончательно новый весовой вектор W_n является отрезком, направленным из начала координат в конец вектора δ . Отсюда можно видеть, что эффект обучения состоит во вращении весового вектора в направлении входного вектора без существенного изменения его длины.



Переменная α является коэффициентом скорости обучения, который вначале обычно равен $\sim 0,7$ и может постепенно уменьшаться в процессе обучения. Это позволяет делать большие начальные шаги для быстрого грубого обучения и меньшие шаги при подходе к окончательной величине.

Если бы с каждым нейроном Кохонена ассоциировался один входной вектор, то слой Кохонена мог бы быть обучен с помощью одного вычисления на вес. Веса нейрона-победителя приравнивались бы к компонентам обучающего вектора ($\alpha = 1$). Как правило, обучающее множество включает много сходных между собой входных векторов, и сеть должна быть обучена активировать один и тот же нейрон Кохонена для каждого из них. В этом случае веса этого нейрона должны получаться усреднением входных векторов, которые должны его активировать. Постепенное уменьшение величины α уменьшает воздействие каждого обучающего шага, так что окончательное значение будет средней величиной от входных векторов, на которых происходит обучение. Таким образом, веса, ассоциированные с нейроном, примут значение вблизи «центра» входных векторов, для которых данный нейрон является «победителем».

Всем весам сети перед началом обучения следует придать начальные зна-

чения. Общепринятой практикой при работе с нейронными сетями является присваивание весам небольших случайных значений. При обучении слоя Кохонена случайно выбранные весовые векторы следует нормализовать. Окончательные значения весовых векторов после обучения совпадают с нормализованными входными векторами. Поэтому нормализация перед началом обучения приближает весовые векторы к их окончательным значениям, сокращая, таким образом, обучающий процесс.

Рандомизация весов слоя Кохонена может породить серьезные проблемы при обучении, так как в результате ее весовые векторы распределяются равномерно по поверхности гиперсферы. Из-за того, что входные векторы, как правило, распределены неравномерно и имеют тенденцию группироваться на относительно малой части поверхности гиперсферы, большинство весовых векторов будут так удалены от любого входного вектора, что они никогда не будут давать наилучшего соответствия. Эти нейроны Кохонена будут всегда иметь нулевой выход и окажутся бесполезными. Более того, оставшихся весов, дающих наилучшие соответствия, может оказаться слишком мало, чтобы разделить входные векторы на классы, которые расположены близко друг к другу на поверхности гиперсферы.

Допустим, что имеется несколько множеств входных векторов, все множества сходные, но должны быть разделены на различные классы. Сеть должна быть обучена активировать отдельный нейрон Кохонена для каждого класса. Если начальная плотность весовых векторов в окрестности обучающих векторов слишком мала, то может оказаться невозможным разделить сходные классы из-за того, что не будет достаточного количества весовых векторов в интересующей нас окрестности, чтобы приписать по одному из них каждому классу входных векторов.

Наоборот, если несколько входных векторов получены незначительными изменениями из одного и того же образца и должны быть объединены в один класс, то они должны включать один и тот же нейрон Кохонена. Если же плот-

ность весовых векторов очень высока вблизи группы слегка различных входных векторов, то каждый входной вектор может активировать отдельный нейрон Кохонена.

Задание

Применить нейронную сеть Кохонена с самообучением для задачи кластеризации. На первом этапе сгенерировать случайные точки на плоскости вокруг 2 центров кластеризации (примерно по 20-30 точек). Далее считать, что сеть имеет два входа (координаты точек) и два выхода – один из них равен 1, другой 0 (по тому, к какому кластеру принадлежит точка). Подавая последовательно на вход (вразнобой) точки, настроить сеть путем применения описанной процедуры обучения так, чтобы она приобрела способность определять, к какому кластеру принадлежит точка. Коэффициент α выбрать, уменьшая его от шага к шагу по правилу $\alpha = (50-i)/100$, причем для каждого нейрона это будет своё значение, а подстраиваться на каждом шаге будут веса только одного (выигравшего) нейрона.

Контрольные вопросы

1. Что такое нейронная сеть Кохонена?
2. В чем заключается алгоритм обучения нейронной сети без учителя? До каких пор проводится обучение?
3. Что такое скорость обучения? В каких пределах ее можно менять?
4. Возможно ли применение обучения с учителем к сетям Кохонена? Обосновать ответ.