Методические рекомендации по дисциплине «Основы нейрокибернетики»

Лабораторные работы 1-8. Разработка экспертной системы в среде VISUAL PROLOG

Экспертные системы (ЭС) - это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и тиражирующие этот эмпирический опыт для консультаций менее квалифицированных пользователей.

Построим небольшую экспертную систему, которая будет определять одну из нескольких рыб по признакам, указанным пользователем. Система будет задавать вопросы и строить логические выводы на основе полученных ответов.

Типичный диалог экспертной системы с пользователем может выглядеть следующим образом (рис.27):

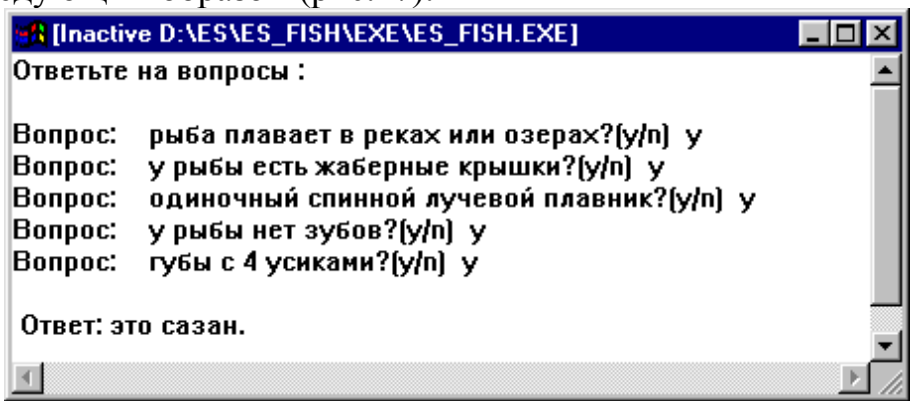


рис.27. Диалог экспертной системы с пользователем

Первым шагом построения такой системы является обеспечение ее знаниями, необходимыми для выполнения рассуждений. Программа должна во время консультаций выводить заключения из информации, имеющейся в базе знаний, а также использовать новую информацию, полученную от пользователя. Поэтому минимальная ЭС должна включать:

- базу знаний;
- механизм вывода;
- пользовательский интерфейс.

Разработку любой ЭС следует начать с исследования предметной области. Пусть на основе бесед с экспертом были получены следующие эмпирические правила:

1) ЕСЛИ
   это отряд карпообразные и
   И
   у рыбы желто-золотистый окрас
   И
   губы с 4 усиками
   ТО

это сазан

2) ЕСЛИ
это отряд карпообразные
И
у рыбы плавники с розовыми перьями
ТО
это плотва
3) ЕСЛИ
спинной плавник узкий
И
у рыбы желто-золотистый окрас
И
это отряд карпообразные
ТО
это лещ
4) ЕСЛИ
у рыбы нет зубов
И
одиночный спинной лучевой плавник
И
это костная рыба
И
это пресноводная рыба
ТО
это отряд карпообразные
5) ЕСЛИ
у рыбы есть костный скелет
ИЛИ
у рыбы есть жаберные крышки
ТО
это костная рыба
6) ЕСЛИ
рыба плавает в озерах
ИЛИ
рыба плавает в реках
ТО
это пресноводная рыба

Для создания базы знаний используем предикаты:

```
fish(symbol)
otrajd(symbol)
vid(symbol)
priznak(symbol)
```

Базу знаний будут составлять следующие правила:

```
fish("это сазан"):-
          otrajd("отряд карпообразные"),
```

```
        priznak("губы с 4 усиками").
fish("это плотва"):-
        otrajd("отряд карпообразные"),
        priznak("плавники с розовыми перьями").
fish("это лещ"):-
        otrajd("отряд карпообразные"),
        priznak("у рыбы желто-золотистый окрас"),
        priznak("у рыбы спинной плавник узкий").
```

Необходимо предусмотреть, что искомой рыбы в базе знаний нет:

```
fish("Данной рыбы в базе знаний не обнаружено").
otrajd("отряд карпообразные"):-
        vid("пресноводная рыба"),
        vid("костная рыба"),
        priznak("одиночный спинной лучевой плавник"),
        priznak("у рыбы нет зубов").
vid("костная рыба"):-
        priznak("у рыбы есть жаберные крышки");
        priznak("у рыбы есть костный скелет").
vid("пресноводная рыба"):-
        priznak("рыба плавает в реках или озерах").
```

Для хранения информации, полученной от пользователя, используются предикаты **yes** и **no**, составляющие внутреннюю базу фактов. Предикат **yes** служит для хранения фактов, соответствующих положительному ответу, а предикат **no** – для хранения отрицательных ответов. Т.е. предикат **yes** утверждает наличие какого-либо признака у рыбы, а **no** – отсутствие указанного признака. Эти предикаты объявляются в разделе внутренней базы фактов:

```
global facts
    yes (symbol)
    no  (symbol)
```

Добавить новые факты во внутреннюю базу можно с помощью правила add_to_database, состоящего из двух частей. Первая часть добавляет факты, соответствующие положительному ответу (с клавиатуры вводится ‘y’). Вторая часть правила добавляет факты, указывающие на отсутствие данного признака у рыбы.

```
add_to_database (Y,'y') :- assertz (yes (Y)).
add_to_database (Y,'n') :- assertz (no (Y)),fail.
```

Необходимо предусмотреть очистку внутренней базы фактов. Для этого создадим правило:

```
clear_from_database :- retract(yes(_)),fail.
clear_from_database :- retract(no(_)),fail.
```

Для проверки наличия у рыбы определенного признака создадим правило priznak (Y):

```
priznak (Y) :-  yes (Y),!.
priznak (Y) :-  not(no (Y)),
                question (Y).
```

Формулировка вопроса, ввод ответа и сохранение соответствующего правила осуществляется с помощью правил:

```
answer :- fish(X),!,nl,
          save("BF1.dbf"),
          write (" Ответ: ",X,"."),nl.
question(Y) :-
          write ("Вопрос:    ",Y,"?(y/n)   "),
          otvet(X),
          write(X),nl,
          add_to_database (Y,X).
otvet(C):-readchar(C).
```

И, наконец, правило begin, запускающее сеанс консультации:

```
begin :- write ("Ответьте на вопросы :"),nl,nl,
          answer,
          clear_from_database,
          nl,nl,nl,nl,
          exit.
```

Полный листинг программы выглядит следующим образом:

```
GLOBAL FACTS
    yes (symbol)
    no  (symbol)
PREDICATES
    fish(symbol)
    otrajd(symbol)
    vid(symbol)
```

```
    begin
    answer
    question(symbol)
    add_to_database(symbol,char)
    otvet(char)
    clear_from_database
    priznak(symbol)
GOAL
    begin.
CLAUSES
    begin :-
        write ("Ответьте на вопросы :"),nl,nl,
        answer,
        clear_from_database,
        nl,nl,nl,nl,
        exit.
    answer :-
        fish(X),!,nl,
        save("BF1.dbf"),
        write (" Ответ: ",X,"."),nl.
    question(Y) :-
        write ("Вопрос:    ",Y,"? "),
        otvet(X),
        write(X),nl,
        add_to_database (Y,X).
    otvet(C):-
        readchar(C).
    priznak (Y) :-
        yes (Y),!.
    priznak (Y) :-
        not( no (Y)),
        question (Y).
    add_to_database (Y,'y') :-
```

```
        assertz (yes (Y)).
    add_to_database (Y,'n') :-
        assertz (no (Y)),fail.
    clear_from_database :- retract (yes(_)),fail.
    clear_from_database :- retract (no(_)),fail.
    fish("это сазан"):-
        otrajd("отряд карпообразные"),
        priznak("губы с 4 усиками").
    fish("это плотва"):-
        otrajd("отряд карпообразные"),
        priznak("плавники с розовыми перьями").
    fish("это лещ"):-
        otrajd("отряд карпообразные"),
        priznak("у рыбы желто-золотистый окрас"),
        priznak("у рыбы спинной плавник узкий").
    fish("Данной рыбы в базе знаний не обнаружено").
    otrajd("отряд карпообразные"):-
        vid("пресноводная рыба"),
        vid("костная рыба"),
        priznak("одиночный спинной лучевой плавник"),
        priznak("у рыбы нет зубов").
    vid("костная рыба"):-
        priznak("у рыбы есть жаберные крышки");
        priznak("у рыбы есть костный скелет").
    vid("пресноводная рыба"):-
        priznak(«рыба плавает в реках или озерах»).
```

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Реализуйте данную программу в среде Visual Prolog и протестируйте ее.

2. Расширьте базу знаний экспертной системы, добавив следующие правила:

1) ЕСЛИ

у рыбы есть электрические органы

И

это отряд скаты

ТО

это электрический скат

2) ЕСЛИ

у рыбы на хвосте ядовитый шип

И

это отряд скаты

ТО

это скат-хвостокол

3) ЕСЛИ

у рыбы серо-коричневый окрас

И

у рыбы коническая морда

И

это отряд акулы

ТО

это гиганская акула

4) ЕСЛИ

это отряд акулы

И

рыба нападает на людей

И

у рыбы молотообразная морда

ТО

это рыба молот

5) ЕСЛИ

у рыбы нет хвостового плавника

И

у рыбы тонкий длинный хвост

И

это хрящевая рыба

И

это морская рыба

ТО

это отряд скаты

6) ЕСЛИ

это морская рыба

И

это хрящевая рыба

И

плавники не гибкие

И

хвост ассиметричный

ТО

это отряд акулы

7) ЕСЛИ

у рыбы нет плавательного пузыря

ИЛИ

у рыбы есть хрящевый скелет

ТО

это хрящевая рыба

8) ЕСЛИ

рыба плавает в морях

ТО

это морская рыба

3. Протестируйте полученную экспертную систему.

Варианты	**Задание**
1	Построить систему, которая дает возможность распознавать птиц
2	Построить систему, которая дает возможность распознавать время года.
3	Построить систему, которая дает возможность распознавать фрукты
4	Построить систему, которая дает возможность распознавать виды транспорта

Лабораторные работы 9 - 16. Разработка и обучение нейросети средствами объектно-ориентированного программирования

Нейросеть – это обучаемая система. Она действует не только в соответствии с заданным алгоритмом и формулами, но и на основании прошлого опыта. Этакий ребенок, который с каждым разом складывает пазл, делая все меньше ошибок.
И, как принято писать у модных авторов – нейросеть состоит из нейронов.
Тут нужно сделать остановку и разобраться.

Договоримся, что нейрон – это просто некая воображаемая чёрная коробка, у которой кучка входных отверстий и одно выходное.
Причем как входящая, так и исходящая информация может быть аналоговой (чаще всего так и будет).
Как выходной сигнал формируется из кучи входных – определяет внутренний алгоритм нейрона.
Для примера напишем небольшую программу, которая будет распознавать простые изображения, скажем, буквы русского языка на растровых изображениях.
Условимся, что в исходном состоянии наша система будет иметь «пустую» память, т.е. этакий новорожденный мозг, готовый к бою.
Для того чтобы заставить его корректно работать, нам нужно будет потратить время на обучение.
Уворачиваясь от летящих в меня помидоров, скажу, что писать будем на Delphi (на момент написания статьи была под рукой). Если возникнет необходимость – помогу перевести пример на другие языки.
Также прошу легкомысленно отнестись к качеству кода – программа писалась за час, просто чтобы разобраться с темой, для серьезных задач такой код вряд ли применим.
Итак, исходя из поставленной задачи — сколько вариантов выхода может быть? Правильно, столько, сколько букв мы будем уметь определять. В алфавите их пока только 33, на том и остановимся.
Далее, определимся со входными данными.Чтобы слишком не заморачиватсья – будем подавать на вход битовый массив 30x30 в виде растрового изображения:

В итоге – нужно создать 33 нейрона, у каждого из которых будет 30x30=900 входов.
Создадим класс для нашего нейрона:

```
type
  Neuron = class
    name: string; // Тут название нейрона – буква, с которой он ассоциируется
    input: array[0..29,0..29] of integer; // Тут входной массив 30x30
    output:integer; // Сюда он будет говорить, что решил
    memory:array[0..29,0..29] of integer; // Тут он будет хранить опыт о предыдущем опыте
  end;
```

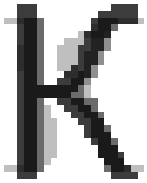
Создадим массив нейронов, по количеству букв:

```
For i:=0 to 32 do begin
neuro_web[i]:=Neuron.Create;
neuro_web[i].output:=0; //  Пусть пока молчит
 neuro_web[i].name:=chr(Ord('A')+i); // Буквы от А до Я
end;
```

Теперь вопрос – где мы будем хранить «память» нейросети, когда программа не работает?

Чтобы не углубляться в INI или, не дай бог, базы данных, я решил хранить их в тех же растровых изображениях 30x30.

Вот например, память нейрона «К» после прогона программы по разным шрифтам:



Как видно, самые насыщенные области соответствуют наиболее часто встречаемым пикселям.
Будем загружать «память» в каждый нейрон при его создании:

```
p:=TBitmap.Create;
    p.LoadFromFile(ExtractFilePath(Application.ExeName)+'\res\'+
neuro_web[i].name+'.bmp')
```

В начале работы необученной программы, память каждого нейрона будет белым пятном 30x30.
Распознавать нейрон будет так:
— Берем 1й пиксель
— Сравниваем его с 1м пикселем в памяти (там лежит значение 0..255)
— Сравниваем разницу с неким порогом
— Если разница меньше порога – считаем, что в данной точке буква похожа на лежащую в памяти, добавляем +1 к весу нейрона.
И так по всем пикселям.
Вес нейрона – это некоторое число (в теории до 900), которое определяется степенью сходства обработанной информации с хранимой в памяти.
В конце распознавания у нас будет набор нейронов, каждый из которых считает, что он прав на сколько-то процентов. Эти проценты – и есть вес нейрона. Чем больше вес, тем вероятнее, что именно этот нейрон прав.
Теперь будем скармливать программе произвольное изображение и пробегать каждым нейроном по нему:

```
for x:=0 to 29 do
        for y:=0 to 29 do begin
            n:=neuro_web[i].memory[x,y];
            m:=neuro_web[i].input[x,y];
```

```
if ((abs(m-n)<120)) then // Порог разницы цвета
if  m<250  then    neuro_web[i].weight:=neuro_web[i].weight+1;  //
Кроме того, не будем учитывать белые пиксели, чтобы не получать
лишних баллов в весах
if m<>0 then   begin
if m<250 then   n:=round((n+(n+m)/2)/2);
neuro_web[i].memory[x,y]:=n;      end
else if n<>0 then
if m<250 then    n:=round((n+(n+m)/2)/2);
neuro_web[i].memory[x,y]:=n;
end;
```

Как только закончится цикл для последнего нейрона – будем выбирать из всех тот, у которого вес больше:

```
if neuro_web[i].weight>max then  begin
max:=neuro_web[i].weight;
max_n:=i;
end;
```

Именно по вот этому значению max_n, программа и скажет нам, что, по её мнению, мы ей подсунули.

По началу это будет не всегда верно, поэтому нужно сделать алгоритм обучения.

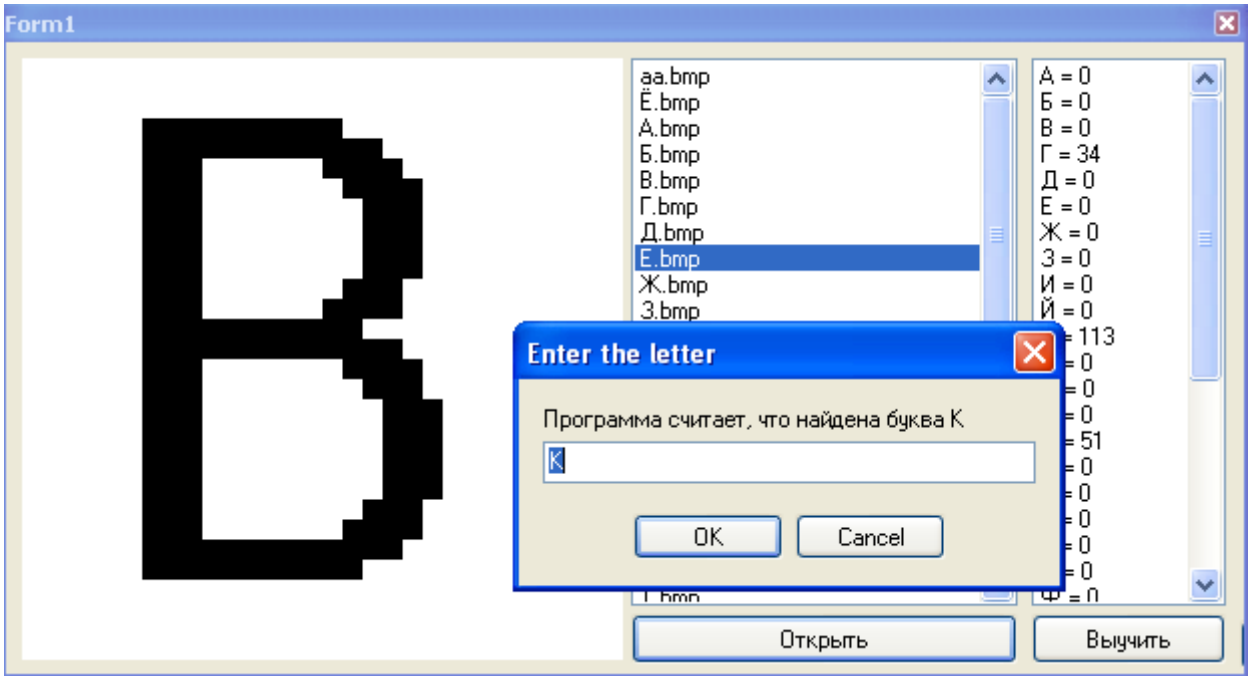
```
s:=InputBox('Enter the letter', ‘программа считает, что это буква ’+neuro_web[max_n].name, neuro_web[max_n].name);
    for i:=0 to 32 do     begin //Пробегаем по нейронам
    if neuro_web[i].name=s then begin //В нужном нейроне обновляем память
for x:=0 to 29 do   begin
   for y:=0 to 29 do       begin

p.Canvas.Pixels[x,y]:=RGB(neuro_web[i].memory[x,y],neuro_web[i].memory[x,y], neuro_web[i].memory[x,y]); //Записываем новое значение пикселя памяти
    end;
     end;
      p.SaveToFile(ExtractFilePath(Application.ExeName)+'\res\'+ neuro_web[i].name+'.bmp');
```

Само обновление памяти будем делать так:

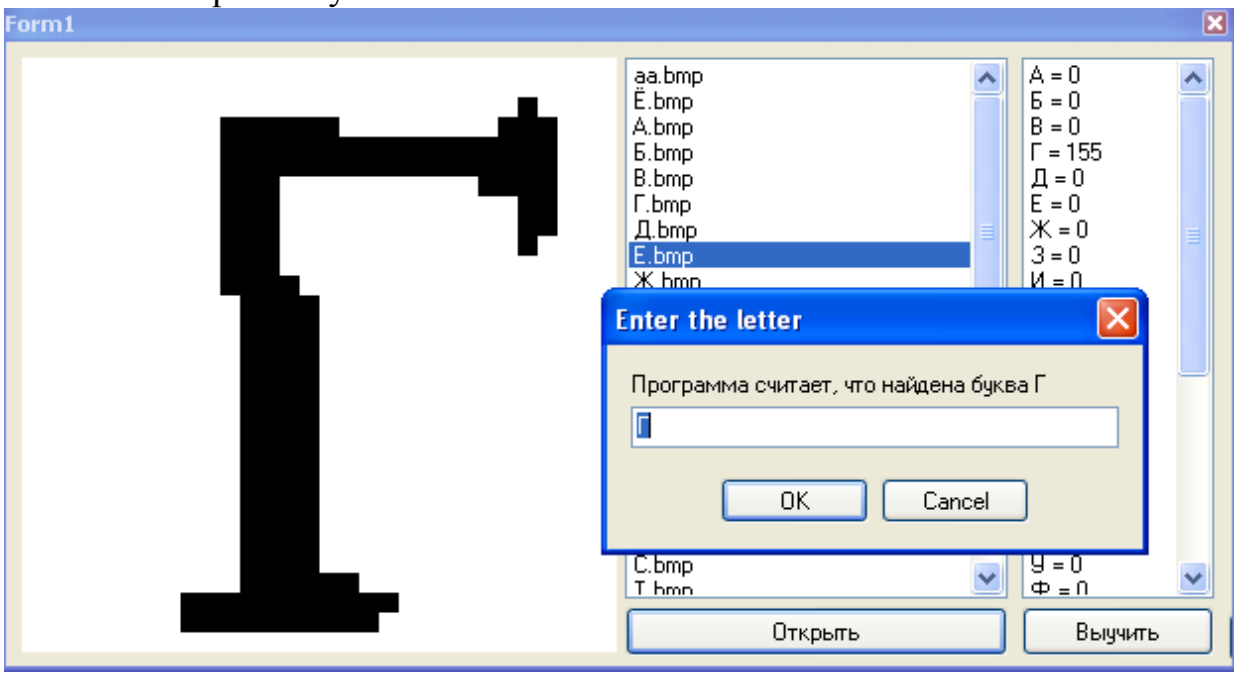
```
n:=round(n+(n+m)/2);
```

Т.е. если данная точка в памяти нейрона отсутствует, но учитель говорит, что она есть в этой букве – мы её запоминаем, но не полностью, а только наполовину. С дальнейшим обучением, степень влияния данного урока будет увеличиваться.
Вот несколько итераций для буквы Г:

На этом наша программа готова.
Обучение
Начнем обучение.
Открываем изображения букв и терпеливо указываем программе на её ошибки:



Через некоторое время программа начнет стабильно определять даже не знакомые ей ранее буквы:



Задание: **Разработать нейросеть распознавания графического образа.**
Темы: Смайлики, Иконки, Эмоции и т.д. по согласованию с преподавателем.