МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ  
  
Федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королёва»

(Самарский университет)  
  
Факультет информатики  
Кафедра программных систем  
  
Дисциплина  
**Системы искусственного интеллекта  
  
  
  
ОТЧЕТ**по лабораторной работе №4  
«Модель Персептрона Розенблатта».

Студент: Гижевская В.Д.   
Группа: 6313-020302D   
  
Преподаватель: Додонов М.В.   
  
Дата:

Самара 2020

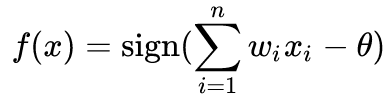
1. Краткая теория

Элементарный персептрон состоит из элементов трёх типов: S-элементов, A-элементов и одного R-элемента. S-элементы — это слой сенсоров или рецепторов. В физическом воплощении они соответствуют, например, светочувствительным клеткам сетчатки глаза или фоторезисторам матрицы камеры. Каждый рецептор может находиться в одном из двух состояний — покоя или возбуждения, и только в последнем случае он передаёт единичный сигнал в следующий слой, ассоциативным элементам.

A-элементы называются ассоциативными, потому что каждому такому элементу, как правило, соответствует целый набор (ассоциация) S-элементов. A-элемент активизируется, как только количество сигналов от S-элементов на его входе превысило некоторую величину θ. Таким образом, если набор соответствующих S-элементов располагается на сенсорном поле в форме буквы «Д», A-элемент активизируется, если достаточное количество рецепторов сообщило о появлении «белого пятна света» в их окрестности, то есть A-элемент будет как бы ассоциирован с наличием/отсутствием буквы «Д» в некоторой области.

Сигналы от возбудившихся A-элементов, в свою очередь, передаются в сумматор R, причём сигнал от i-го ассоциативного элемента передаётся с коэффициентом wi. Этот коэффициент называется весом A—R связи.

Так же как и A-элементы, R-элемент подсчитывает сумму значений входных сигналов, помноженных на веса (линейную форму). R-элемент, а вместе с ним и элементарный персептрон, выдаёт «1», если линейная форма превышает порог θ, иначе на выходе будет «−1». Математически, функцию, реализуемую R-элементом, можно записать так:



Обучение элементарного персептрона состоит в изменении весовых коэффициентов wi связей A—R. Веса связей S—A (которые могут принимать значения {−1; 0; +1}) и значения порогов A-элементов выбираются случайным образом в самом начале и затем не изменяются.

После обучения персептрон готов работать в режиме распознавания или обобщения. В этом режиме персептрону предъявляются ранее неизвестные ему объекты, и персептрон должен установить, к какому классу они принадлежат. Работа персептрон состоит в следующем: при предъявлении объекта возбудившиеся A-элементы передают сигнал R-элементу, равный сумме соответствующих коэффициентов wi. Если эта сумма положительна, то принимается решение, что данный объект принадлежит к первому классу, а если она отрицательна — то ко второму.

1. Задание

Реализуйте алгоритм обучения простой нейронной сети – однослойный персептрон. Проведите исследование зависимости скорости обучения от темпа и начального значения весов. Объясните полученные вами результаты

Нейронной сети предъявляется вектор, состоящий из 10 компонент, каждая из которых может быть нулём или единицей. Сеть должна научиться определять, чего больше - нулей или единиц.

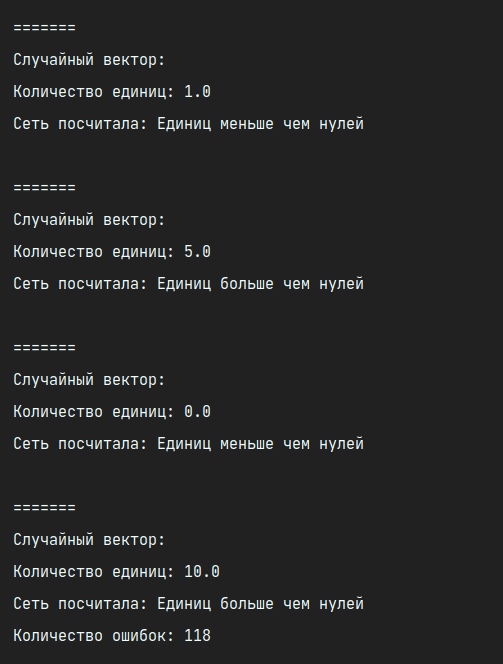


Рисунок 1 – Результат работы программы

1. Листинг программы
2. **class** Main {
4. **public** **static** **void** main(String[] args){
5. **new** Perceptron().call();
6. }
8. }
10. **public** **class** Perceptron {
12. **private** **static** **final** **int** BIAS = 5; *//Граница разделения на true/false*
14. **double** LEARNING\_RATE = 0.001; *//Скорость обучения*
15. **int** NUM\_INSTANCES = 100;  *// колво тестовых векторов*
17. **int** EPOCHS = 50; *//Количество эпох обучения*
19. **double**[] weights = **new** **double**[2]; *// 1 вес для переменной и 1 вес для границы*
21. **double** localError = 0;
22. **int** errorsCount = 0;
24. **double**[] x = **new** **double** [NUM\_INSTANCES];
25. **int**[] outputs = **new** **int** [NUM\_INSTANCES];
27. **public** **void** call(){
28. *// вариации, где единиц < нулей*
29. **for**(**int** i = 0; i < NUM\_INSTANCES/2; i++){
30. x[i] = randomNumber(5 , 10);
31. outputs[i] = 1;
32. }
34. *//вариации, где единиц > нулей*
35. **for**(**int** i = 50; i < NUM\_INSTANCES; i++){
36. x[i] = randomNumber(0 , 4);
37. outputs[i] = 0;
38. }
40. **int** output;
42. weights[0] = randomNumber(0, 1);*// вес*
43. weights[1] = randomNumber(0, 1);*// вес границы*
45. **for** (**int** i = 0; i < EPOCHS; i++) {
46. **for** (**int** p = 0; p < NUM\_INSTANCES; p++) {
47. output = calculateOutput(weights, x[p]);
48. localError = outputs[p] - output; *// разница между ожидаемым и поступившим значениями*
49. weights[0] += LEARNING\_RATE \* localError \* x[p];  *//обновляем веса*
50. weights[1] += LEARNING\_RATE \* localError;
51. }
52. }
54. **for**(**int** j = 0; j < 30; j++){
55. **double** x1 = randomNumber(0 , 10);
57. output = calculateOutput(weights, x1);
59. String out;
60. **if**(output == 1){
61. out = "Единиц больше чем нулей";
62. }
63. **else** {
64. out = "Единиц меньше чем нулей";
65. }
67. System.out.println("**\n**=======**\n**Случайный вектор:");
68. System.out.println("Количество единиц: " + x1);
69. System.out.println("Сеть посчитала: " + out);
70. }
71. System.out.println("Количество ошибок: " + errorsCount);
72. }
74. **private** **double** randomNumber(**int** min , **int** max) {
75. DecimalFormat df = **new** DecimalFormat("#");
76. **return** Double.parseDouble(df.format(min + Math.random() \* (max - min)));
77. }
79. **private** **int** calculateOutput(**double**[] weights, **double** x) { *//сумматор*
80. **double** sum = x \* weights[0] + weights[1];
81. **int** z = (sum >= BIAS) ? 1 : 0;
83. **if**(x >= 5.0){
84. **if**(z == 0) errorsCount++;
85. }
86. **else**{
87. **if**(z == 1) errorsCount++;
88. }
90. **return** z;
91. }
93. }
94. Выводы

Анализируя выходные данные и результаты, полученные на этапе разработки и тестирования программы, можно утверждать, что:

* Точность результата практически не зависит от темпа обучения.
* Точность результата сильно зависит от количества эпох.
* Точность результата сильно зависит от количества входных данных.