Assignment No. 1

The Delta rule

Intelligent Data Processing Laboratory

Paweł Jeziorski, 234066

Mateusz Maciaszek 234087

02.11.2020

1. Main goal

The aim of the task was to implement a single neuron trained with delta rule with the use of single and multiple training patterns. It was then tested to investigate the influence of various parameters on the training process.

2. Theoretical background

Delta rule is a gradient descent learning rule for updating the weights of the artificial neuron to be multiplied by inputs in a single neuron. To adjust the weights, it uses a difference between target and obtained results. The idea of that rule is to find the best fit weight for our training set from space of all possible weights value.

To calculate neuron's value, we use the linear activation function. The output value is simply equal to the sum of the network's respective input/weight products. The linear activation function allows to calculate the derivative of the error.

For any set of inputs and weights, we can determine an associated error value that is measured by the error function or cost function commonly given as the sum of the squares of the differences between each target and actual node activation. The Delta rule will use this function to find the appropriate weight values, i.e., to minimize this error.

After proper simplification of the formulas, the final formula (with a linear activation function) for modifying weights is:

$$\Delta w = lr \bullet (z - y) \bullet x_i,$$

where lr – learning rate, y – neuron output, z- desired output value from pattern, $x_{i\,-}$; i = 1, … ,N – input value.

Additionally, updating comes after each misclassification, so that the chance of reaching the global minimum is high.

3. Experiments and results

3.1 Experiment No. 1

3.2.1. Description

The aim of the experiment is to check the influence of individual parameters on the correctness and effectiveness of the solution.

The considered algorithm has several variables, such as the number of iterations, the learning rate or the number of inputs. They should be initiated in an appropriate and thoughtful way because the effectiveness of the algorithm depends on them.

Three different parameters were used for the tests:

- numbers of iteration (epoch)

- learning rate

- the size of the training set (patterns)

The individual parameter values to be tests are presented in the tables. Orange mark means standard value – the one that stays constant when other parameters are tested.

Table 3.1. Tested values for parameter Numbers of iteration.

Numbers of iteration		
3	10	30

Table 3.2. Tested values for parameter Learning rate.

Learning rate			
0.1	0.01	0.001	0.0001

Table 3.3. Tested values for parameter The size of the training set.

The size of the training set	
20	50

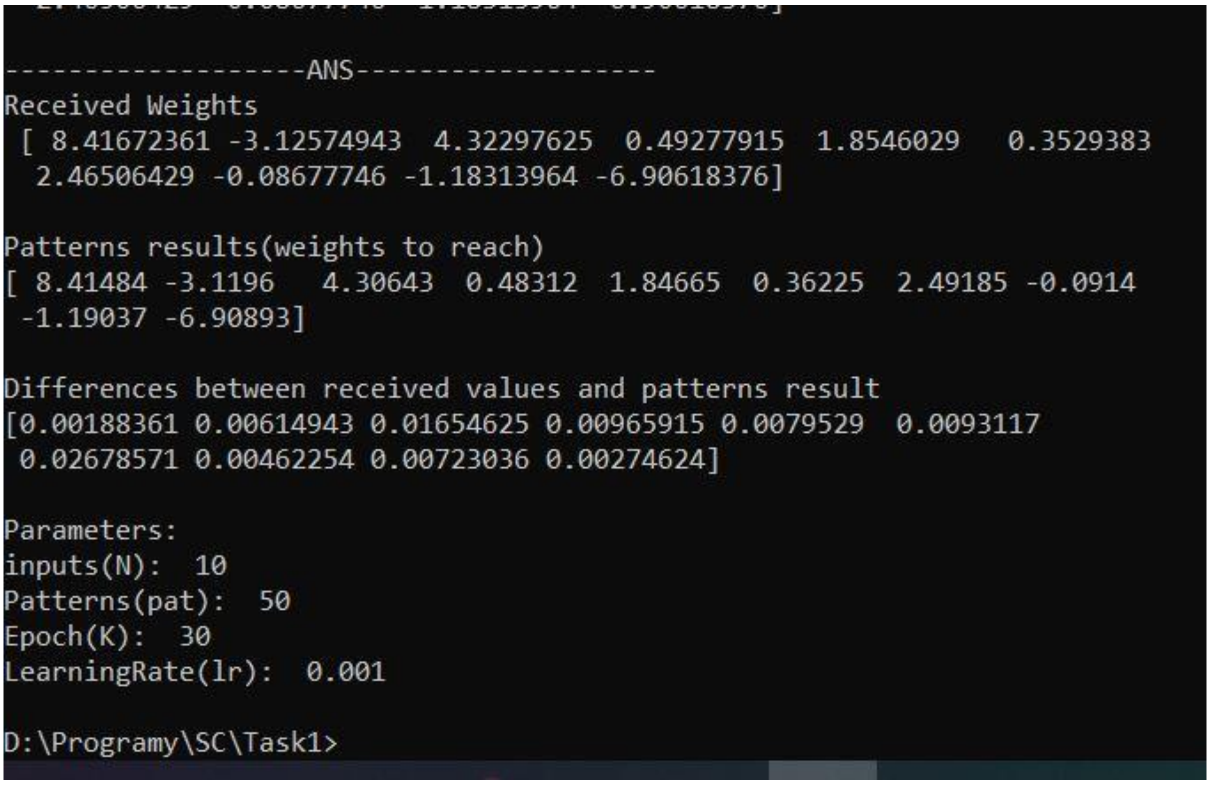
Seven different test cases were obtained after combining each parameter.

Table 3.4. Test Cases

Test case	Numbers of iteration	Learning rate	The size of the training set
1	30	0.001	50
2	3	0.001	50
3	10	0.001	50
4	30	0.1	50
5	30	0.01	50
6	30	0.0001	50
7	30	0.001	20

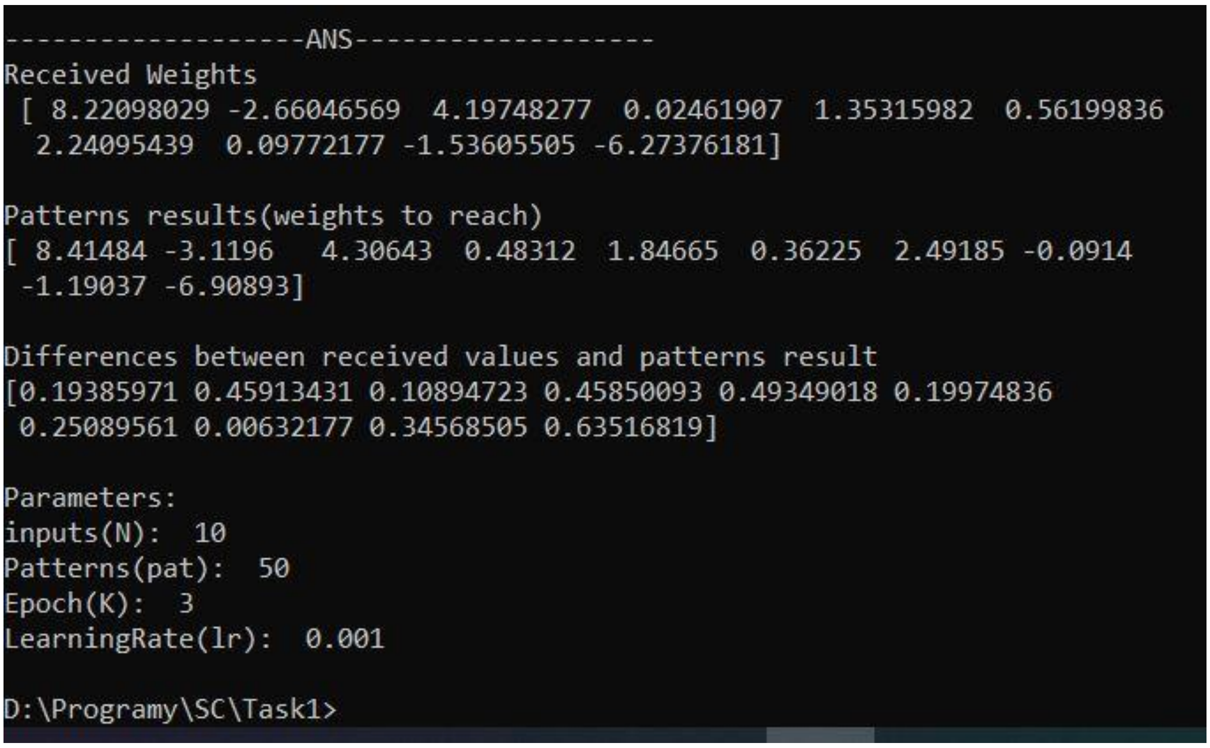
3.2.2. Results

Picture 3.1. Test Case 1



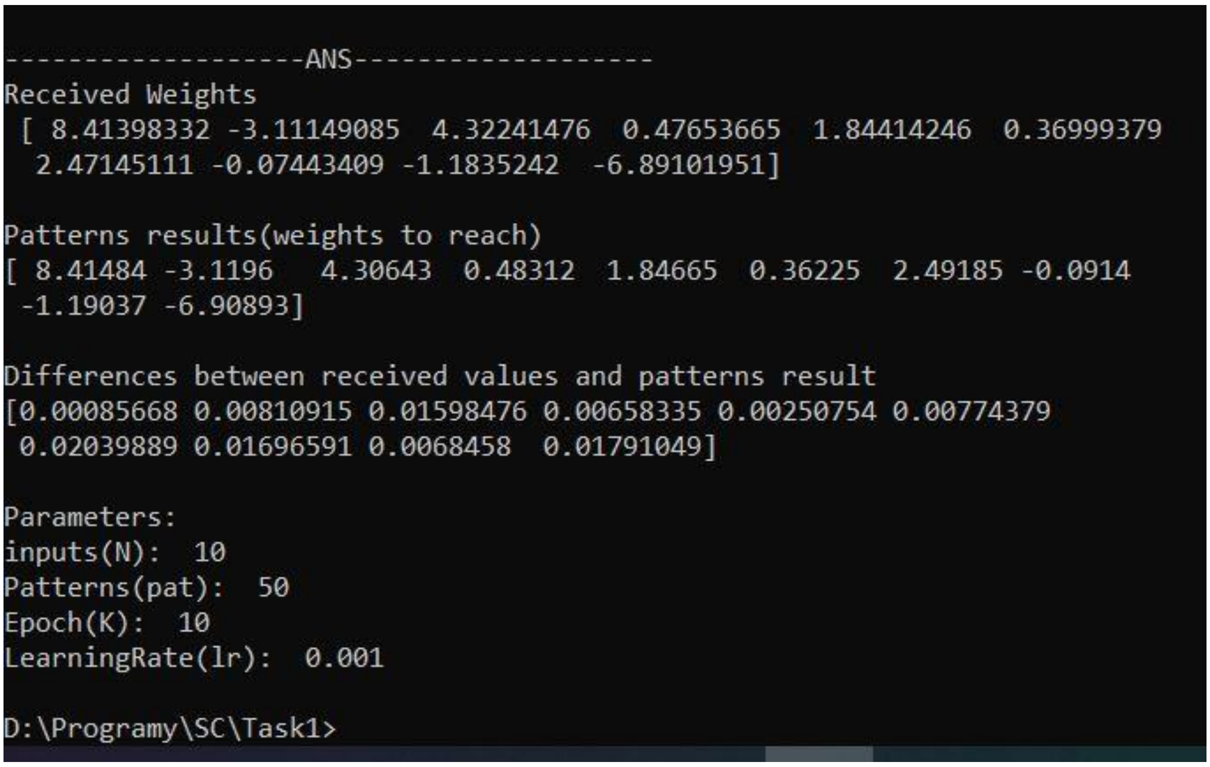
```
-------------------ANS-------------------
Received Weights
 [ 8.41672361 -3.12574943  4.32297625  0.49277915  1.8546029   0.3529383
  2.46506429 -0.08677746 -1.18313964 -6.90618376]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[0.00188361 0.00614943 0.01654625 0.00965915 0.0079529  0.0093117
 0.02678571 0.00462254 0.00723036 0.00274624]

Parameters:
inputs(N):  10
Patterns(pat):  50
Epoch(K):  30
LearningRate(lr):  0.001

D:\Programy\SC\Task1>
```

Picture 3.2. Test Case 2



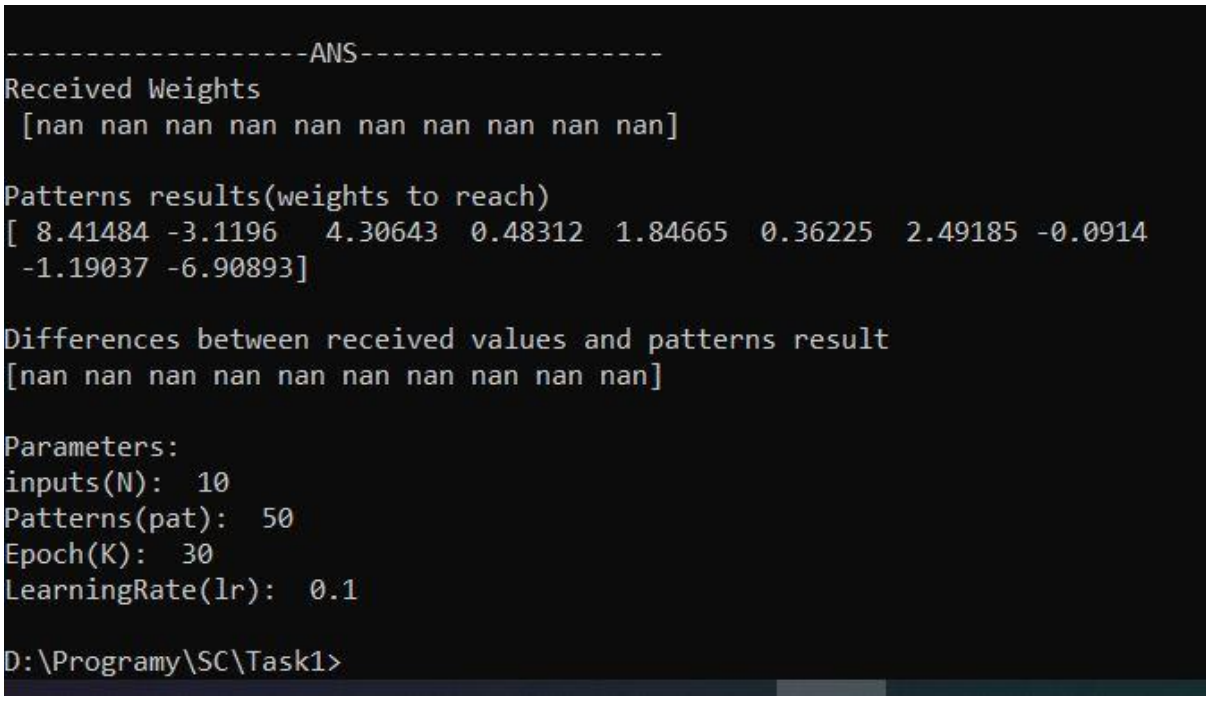
```
-------------------ANS-------------------
Received Weights
 [ 8.22098029 -2.66046569  4.19748277  0.02461907  1.35315982  0.56199836
  2.24095439  0.09772177 -1.53605505 -6.27376181]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[0.19385971 0.45913431 0.10894723 0.45850093 0.49349018 0.19974836
 0.25089561 0.00632177 0.34568505 0.63516819]

Parameters:
inputs(N):  10
Patterns(pat):  50
Epoch(K):  3
LearningRate(lr):  0.001

D:\Programy\SC\Task1>
```

Picture 3.3. Test Case 3



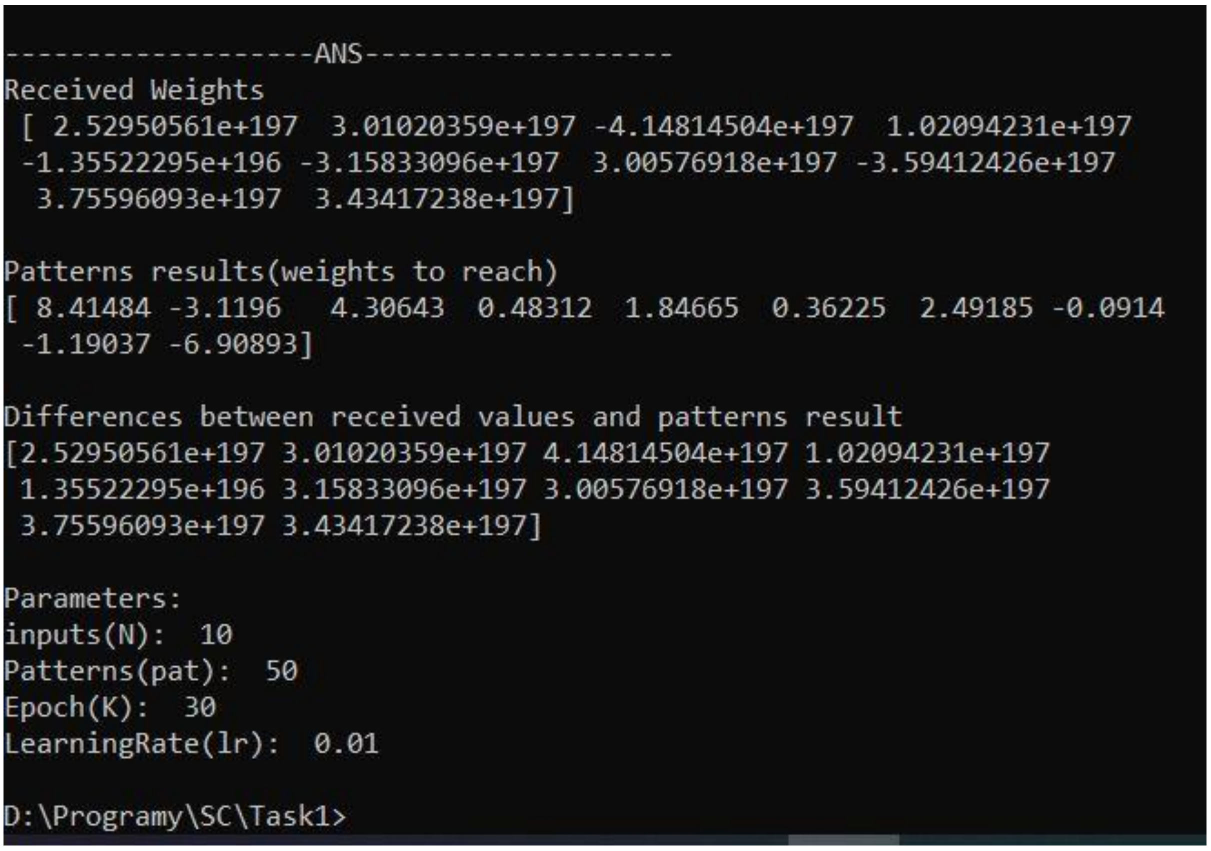
```
-------------------ANS-------------------
Received Weights
 [ 8.41398332 -3.11149085  4.32241476  0.47653665  1.84414246  0.36999379
  2.47145111 -0.07443409 -1.1835242  -6.89101951]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[0.00085668 0.00810915 0.01598476 0.00658335 0.00250754 0.00774379
 0.02039889 0.01696591 0.0068458  0.01791049]

Parameters:
inputs(N):  10
Patterns(pat):  50
Epoch(K):  10
LearningRate(lr):  0.001

D:\Programy\SC\Task1>
```

Picture 3.4. Test Case 4



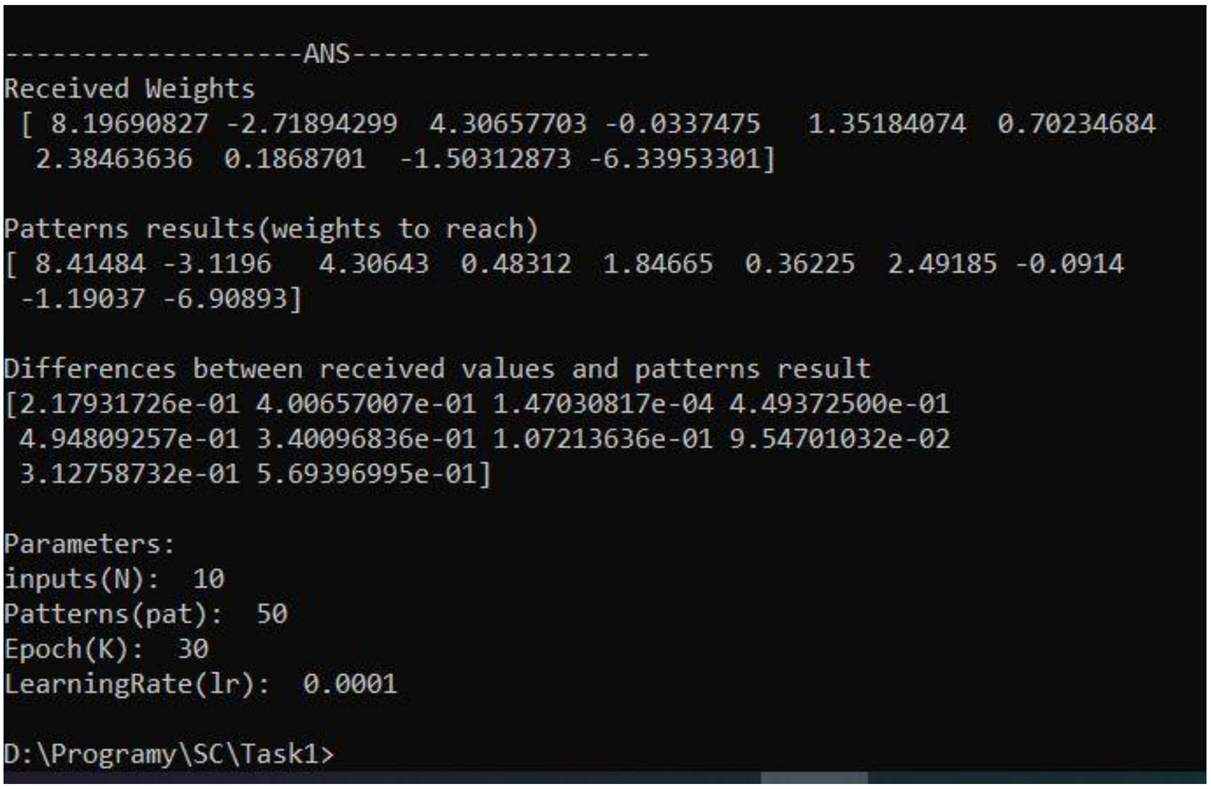
```
-------------------ANS-------------------
Received Weights
 [nan nan nan nan nan nan nan nan nan nan]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[nan nan nan nan nan nan nan nan nan nan]

Parameters:
inputs(N):  10
Patterns(pat):  50
Epoch(K):  30
LearningRate(lr):  0.1

D:\Programy\SC\Task1>
```

Picture 3.5. Test Case 5



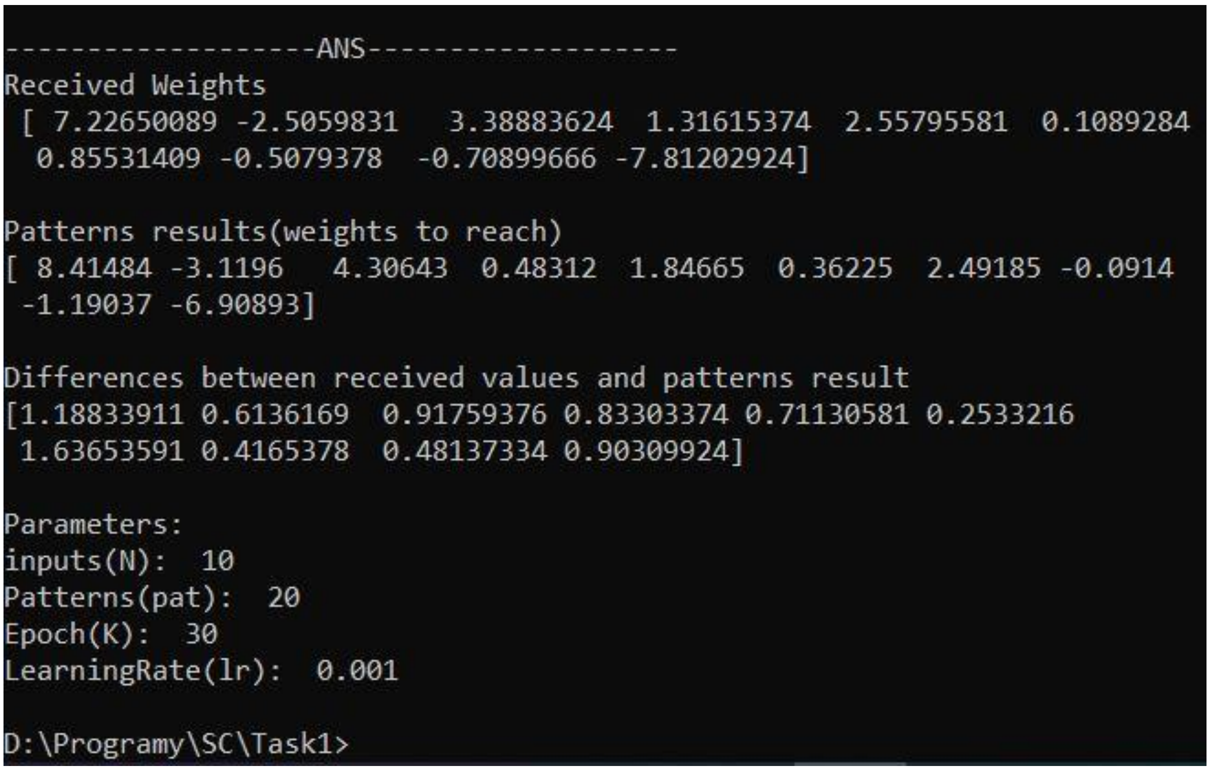
```
-------------------ANS-------------------
Received Weights
 [ 2.52950561e+197  3.01020359e+197 -4.14814504e+197  1.02094231e+197
 -1.35522295e+196 -3.15833096e+197  3.00576918e+197 -3.59412426e+197
  3.75596093e+197  3.43417238e+197]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[2.52950561e+197 3.01020359e+197 4.14814504e+197 1.02094231e+197
 1.35522295e+196 3.15833096e+197 3.00576918e+197 3.59412426e+197
 3.75596093e+197 3.43417238e+197]

Parameters:
inputs(N):  10
Patterns(pat):  50
Epoch(K):  30
LearningRate(lr):  0.01

D:\Programy\SC\Task1>
```

Picture 3.6. Test Case 6



```
-------------------ANS-------------------
Received Weights
 [ 8.19690827 -2.71894299  4.30657703 -0.0337475   1.35184074  0.70234684
  2.38463636  0.1868701  -1.50312873 -6.33953301]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[2.17931726e-01 4.00657007e-01 1.47030817e-04 4.49372500e-01
 4.94809257e-01 3.40096836e-01 1.07213636e-01 9.54701032e-02
 3.12758732e-01 5.69396995e-01]

Parameters:
inputs(N):  10
Patterns(pat):  50
Epoch(K):  30
LearningRate(lr):  0.0001

D:\Programy\SC\Task1>
```

Picture 3.7. Test Case 7



```
-------------------ANS-------------------
Received Weights
 [ 7.22650089 -2.5059831   3.38883624  1.31615374  2.55795581  0.1089284
  0.85531409 -0.5079378  -0.70899666 -7.81202924]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[1.18833911 0.6136169  0.91759376 0.83303374 0.71130581 0.2533216
 1.63653591 0.4165378  0.48137334 0.90309924]

Parameters:
inputs(N):  10
Patterns(pat):  20
Epoch(K):  30
LearningRate(lr):  0.001

D:\Programy\SC\Task1>
```

3.2 Experiment No. 2

3.2.1. Description

The second experiment was designed to show the influence of the number of training patterns fed to the neuron as well as the number of the neuron inputs on the outcomes of the training process.

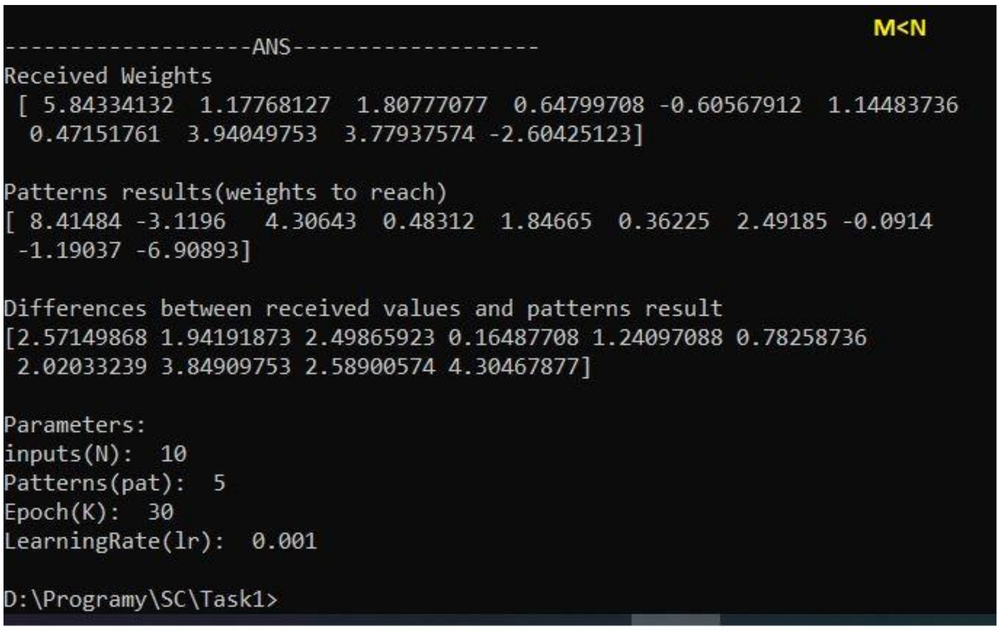
The test cases are shown below.

Table 3.5. Test cases for experiment no.2. M - the number of patterns used, N – the number of inputs.

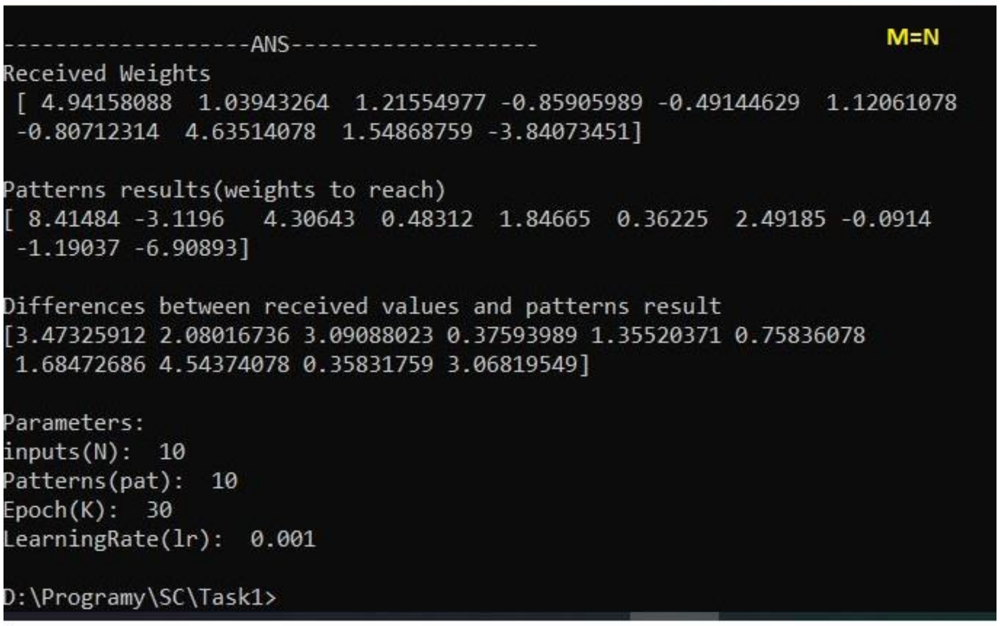
Test cases		
Case	M	N
M<N	5	10
M=N	10	10
M>N	50	10

3.2.2. Results

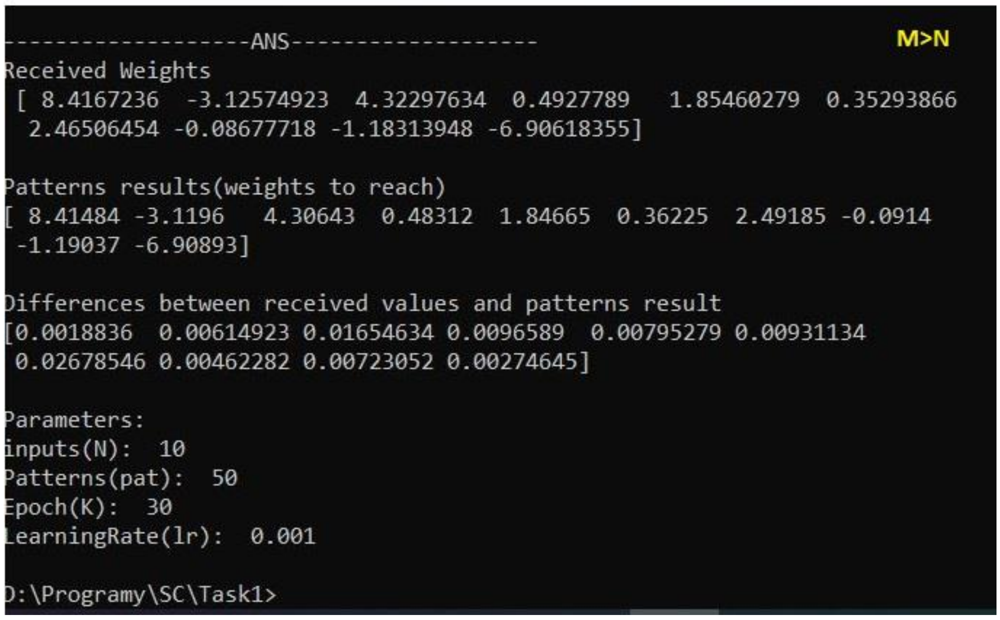
Picture 3.8. The results of the experiment



```
-------------------ANS-------------------
Received Weights
 [ 5.84334132  1.17768127  1.80777077  0.64799708 -0.60567912  1.14483736
  0.47151761  3.94049753  3.77937574 -2.60425123]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[2.57149868 1.94191873 2.49865923 0.16487708 1.24097088 0.78258736
 2.02033239 3.84909753 2.58900574 4.30467877]

Parameters:
inputs(N):  10
Patterns(pat):  5
Epoch(K):  30
LearningRate(lr):  0.001

D:\Programy\SC\Task1>
```



```
-------------------ANS-------------------
Received Weights
 [ 4.94158088  1.03943264  1.21554977 -0.85905989 -0.49144629  1.12061078
 -0.80712314  4.63514078  1.54868759 -3.84073451]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[3.47325912 2.08016736 3.09088023 0.37593989 1.35520371 0.75836078
 1.68472686 4.54374078 0.35831759 3.06819549]

Parameters:
inputs(N):  10
Patterns(pat):  10
Epoch(K):  30
LearningRate(lr):  0.001

D:\Programy\SC\Task1>
```



```
-------------------ANS-------------------
Received Weights
 [ 8.4167236  -3.12574923  4.32297634  0.4927789   1.85460279  0.35293866
  2.46506454 -0.08677718 -1.18313948 -6.90618355]

Patterns results(weights to reach)
[ 8.41484 -3.1196   4.30643  0.48312  1.84665  0.36225  2.49185 -0.0914
 -1.19037 -6.90893]

Differences between received values and patterns result
[0.0018836  0.00614923 0.01654634 0.0096589  0.00795279 0.00931134
 0.02678546 0.00462282 0.00723052 0.00274645]

Parameters:
inputs(N):  10
Patterns(pat):  50
Epoch(K):  30
LearningRate(lr):  0.001

D:\Programy\SC\Task1>
```

4. Summary and conclusions

After performing the experiment 1 the outcomes were the following:

Test Cases 1-2-3 (Numbers of iteration)

For the 3 iterations case, the errors are large. The results obtained are far from what was expected. In the given example, this is not enough iteration to guarantee a correct result. For cases 1 and 3, the result improves significantly. We can see a big, positive difference between case 2 and the others. However, it is worth noting that the accuracy difference between cases 1 and 3 is very small. This may mean that further increasing the iteration may not improve the result.

- Too few iterations will not enable adjusting the neuron weights to the desired values
- Constantly increasing the number of iterations does not improve efficiency (little progress in reaching optimum, a lot of work is required)

Test Cases 4-5-1-6 (Learning Rate)

The fourth case shows behavior in case of too large parameter. The results are being adjusted and modified by a huge value. It doesn't allow to reach the goal, because each adjustment isn't enough gentle to slightly change value.
The lower the value, the more accurate the results. For Test Case 1 they are almost perfect and for Test Case 6 the difference between desired and obtained weight values is almost imperceptible.

- The smaller the value of learning rate, the higher the accuracy.
- The learning rate which is too big makes the algorithm virtually unable to reach the desired weights
- Too small learning rate will make the algorithm reach the desired weight values at a slow pace.

Test Cases 4-7 (The size of the training set)

Limiting the number of patterns does not help finding suitable weights. Too little number of examples makes the algorithm unable to find the best match quickly enough.

- The more patterns, the better the accuracy
- Constantly increasing the number of patterns will burden the algorithm due to the large number of calculations

The results of experiment 2 showed that:

- smaller or equal number of patterns than inputs causes difficulties in obtaining the desired weights
- A large number of training cases guarantees finding the expected result
- Only in case of $M > N$ the correct weights were obtained
- Both $M < N$ and $M = N$ gave very similar results, that were burdened with error
- The greater the number of patterns, the more accurate the results.

To sum up, all the experiments proved, that a relatively simple algorithm enabled find the desired weights of the neuron. Nevertheless, the method has many disadvantages. The delta rule can only be used to train single neuron. As a result, the use of this solution is significantly limited.

An important aspect in every neural network is the selection of parameters. With a deep analysis of the possibilities of the solution used, it is worth carrying out many tests to check the behavior of algorithms for different parameter values. This knowledge allows the algorithm to be optimized. In addition, it is worth providing an appropriate training set that guarantees correct results. According to the tests results, the minimum number of patterns should be greater than the number of inputs.

The delta rule, regardless of its simplicity, was proved to be a good solution for solving the posed problem.