Министерство образования Республики Беларусь Учреждение

образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе

№3 по курсу:

«Модели решения задач в интеллектуальных системах»

Вариант №11

Выполнил студент группы 021702: Семченков Н.А.

Проверил: Жук А.А

МИНСК 2022

1. ЦЕЛЬ

Ознакомиться, проанализировать и получить навыки реализации модели рекуррентной нейронной сети.

2. ПОСТАНОВКА ЗАДАЧИ

Реализовать модель сети Джордана с недовыпрямленной линейной функцией активации (Leaky ReLU).

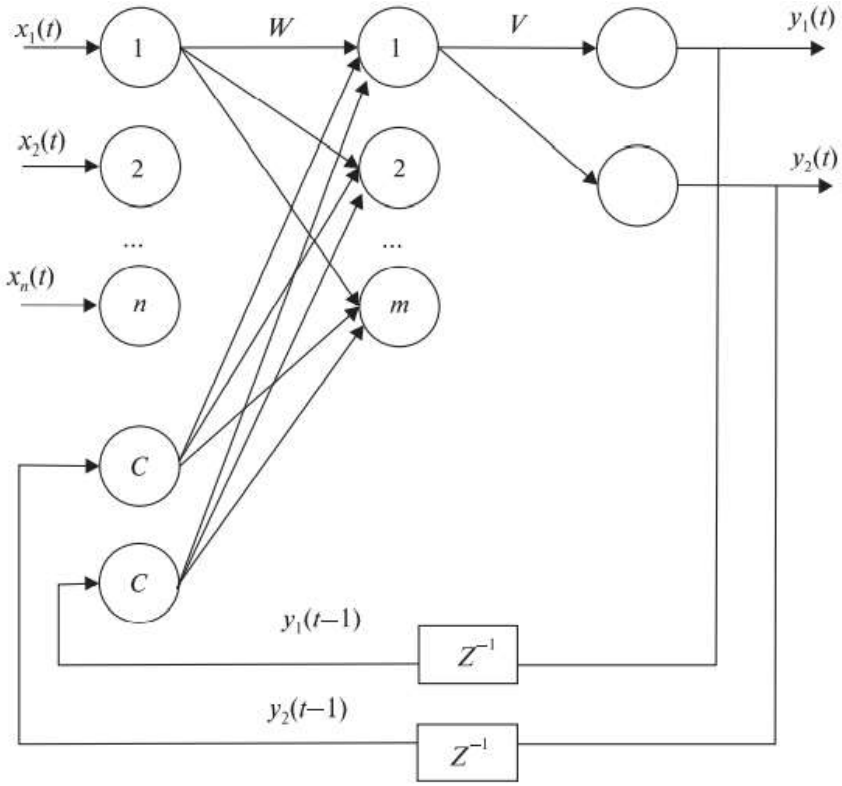
3. ОПИСАНИЕ МОДЕЛИ

Данные:

sequence – исходная последовательность;
resSequence – выходная последовательность;
expSequence – ожидаемая последовательность;
k – размер обучаемой последовательности;
p – количество входных нейронов;
m – количество нейронов на скрытом слое;
e – максимально допустимая ошибка ;
alpha – коэффициент обучения;
N – количество итераций ;
r- количество предсказываемых элементов;

input – входной вектор ;
hidden - вектор скрытого слоя;
output – выходной вектор;
context_output – контекстынй слой ;
X – матрица обучения mxp;
W – матрица весов на скрытом слое pxm;
W_ - матрица весов на выходном слое mx1;
W_C - матрица весов контекстного слоя ;
T - пороговые значения для скрытого слоя;
T_ - пороговые значения для выходного слоя;
expValues – значения, которые необходимо получить при обучении для каждого входного вектора .

Сеть Джордана - рекуррентная сеть в которой выходы нейронных элементов последнего слоя посредством специальных входных нейронов соединены с нейронами промежуточного слоя



Входные нейроны, которые используются для организации обратных связей, называются *контекстными* (context units). Они распределяют выходные данные нейронной сети на нейронные элементы промежуточного слоя (рис. 4.2).

Количество контекстных нейронов равно числу выходных нейронных элементов рекуррентной сети. В качестве выходного слоя таких сетей можно использовать нейроны с линейной функцией активации. Выходное значение j-го нейронного элемента последнего слоя в этом случае вычисляется как

$$y_j(t) = \sum_{i=1}^{m} v_{ij} p_i(t) - T_j, \tag{4.2}$$

где v_{ij} = весовой коэффициент между i-м нейроном промежуточного и j-м нейроном выходного слоя; $p_i(t)$ – выходное значение i-го нейрона промежуточного слоя; T_j – пороговое значение j-го нейрона выходного слоя.

Взвешенная сумма i-го нейрона промежуточного слоя определяется следующим выражением:

$$S_i(t) = \sum_{k=1}^{n} w_{ki} x_k(t) + \sum_{j=1}^{p} w_{ji} y_j(t-1) - T_i\ , \tag{4.3}$$

где w_{ki} – весовой коэффициент между k-м нейроном входного и i-м нейроном скрытого слоя; T_i – пороговое значение i-го нейрона скрытого слоя, n – размерность входного вектора; p – количество нейронов выходного слоя; w_{ji} – весовой коэффициент между j-м контекстным нейроном и i-м нейроном скрытого слоя.

Тогда выходное значение i-го нейрона скрытого слоя равно

$$p_i(t) = F\big(S_i(t)\big). \tag{4.4}$$

В качестве функции активации использовалась недовыпрямленная линейна функция активации

Функция Leaky ReLu – это импровизация обычной функции ReLu. Чтобы решить проблему нулевого градиента для отрицательного значения, Leaky ReLu дает чрезвычайно малую линейную составляющую x отрицательным входам.

Математически:
f (x) = 1 (x < 0);
(αx) +1 (x >= 0) (x).

```
# if x > 0:
#     return x
# else:
#     return alpha * x
```

Производная будет выглядеть так:

```
# if x > 0:
#     return 1
# else:
#     return alpha
```

4. РЕЗУЛЬТАТЫ

Тестирование производилось на трех последовательностях , но так же можно вести свою :

```
1. Fibonacci series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
2. Power function(x^2): 1, 4, 9, 16, 25, 36, 49, 64, 81, ...
3. Sequence of natural numbers: 1, 2, 3, 4, 5, 6, 7, 8, ...
4. Enter your own sequence

Choose sequence:
```

После выбора последовательности, вводиться размер исходной последовательности. Предлагается выбор стандартных параметров, или ввести самому, а также ввести число предсказываемых значений.

Фибоначчи:

```
 Choose sequence: 1
Input number of elements: (7<=n<=15)
15
Learning sequence:
[[  0]
 [  1]
 [  1]
 [  2]
 [  3]
 [  5]
 [  8]
 [ 13]
 [ 21]
 [ 34]
 [ 55]
 [ 89]
 [144]
 [233]
 [377]]
Use standard params(1) or enter your own(2)?:
1
Number of columns in learning matrix p= 2
Number of hidden layer neurons m= 13
Max error e = 0.001
Step learning alfa = 0.001
Max number of learning steps N = 200000
Input number of elements to predict: (1<=n<=10)
7
```

Итог:

```
Iteration:  1  Error:  [30561.30373334]
Iteration:  2  Error:  [14688.1050354]
Iteration:  3  Error:  [34064.36493537]
Iteration:  4  Error:  [14083.00372179]
Iteration:  5  Error:  [21327.01966482]
Iteration:  6  Error:  [20040.99110391]
Iteration:  7  Error:  [18572.24047527]
Iteration:  8  Error:  [16309.44364442]
Iteration:  9  Error:  [6891.02907945]
Iteration:  10  Error:  [2447.60476898]
Finish learning
Iterations =  176
Error =  [0.00099992]
[233 377]
[609.99360966]
[377 610]
[986.99158871]
[610 987]
[1596.98608506]
[ 987 1597]
[2583.97768724]
[1597 2584]
[4180.96378577]
[2584 4181]
[6764.94148649]
[4181 6765]
[10945.90528574]
Result sequence:
Result:  [610.]   Expected value:  610   Line error:  [0.]
Result:  [987.]   Expected value:  987   Line error:  [0.]
Result:  [1597.]   Expected value:  1597   Line error:  [0.]
Result:  [2584.]   Expected value:  2584   Line error:  [0.]
Result:  [4181.]   Expected value:  4181   Line error:  [0.]
Result:  [6765.]   Expected value:  6765   Line error:  [0.]
Result:  [10946.]   Expected value:  10946   Line error:  [0.]
```

Сеть обучилась за 176 итераций.
Полученные результаты близки к эталонным значениям (имеют погрешность), для обучения сети потребовалось небольшое количество итераций для достижения заданной ошибки.

Квадрат чисел:

```
 Choose sequence: 2
Input number of elements: (4<=n<=8)
8
Learning sequence:
[[ 1]
 [ 4]
 [ 9]
 [16]
 [25]
 [36]
 [49]
 [64]]
Use standard params(1) or enter your own(2)?:
1
Number of columns in learning matrix p= 2
Number of hidden layer neurons m= 6
Max error e = 1e-06
Step learning alfa = 0.001
Max number of learning steps N = 100000
Input number of elements to predict: (1<=n<=10)
3
```

Итог:

```
Iteration:  1  Error:  [867.36610388]
Iteration:  2  Error:  [216.62260421]
Iteration:  3  Error:  [101.84567199]
Iteration:  4  Error:  [48.38451234]
Iteration:  5  Error:  [22.88995678]
Iteration:  6  Error:  [11.79556617]
Iteration:  7  Error:  [9.7784563]
Iteration:  8  Error:  [11.01695141]
Iteration:  9  Error:  [11.23459424]
Iteration:  10  Error:  [11.58268256]
Iteration:  10000  Error:  [8.06660485]
Iteration:  20000  Error:  [8.05100194]
Iteration:  30000  Error:  [8.07880093]
Iteration:  40000  Error:  [8.06396828]
Iteration:  50000  Error:  [8.05279756]
Iteration:  60000  Error:  [8.08072271]
Iteration:  70000  Error:  [8.0675401]
Iteration:  80000  Error:  [8.05967391]
Iteration:  90000  Error:  [8.08999384]
Iteration:  100000  Error:  [8.0808741]
Finish learning
Iterations =  100001
Error =  [8.06551666]
[49 64]
[84.47360189]
[64 84]
[111.44811517]
[ 84 111]
[147.1743728]
Result sequence:
Result:  [84.]   Expected value:  81   Line error:  [-3.]
Result:  [111.]   Expected value:  100   Line error:  [-11.]
Result:  [147.]   Expected value:  121   Line error:  [-26.]
```

За 100 тысяч итераций на удалось приблизиться к нужному значению ошибки.
Из полученных результатов только 1-ое спрогнозированное значение близко к эталонному (2-ое и 3-е имеет большую погрешность).

Последовательность натуральных чисел:
Введём параметры сети вручную.

```
 Choose sequence: 3
Input number of elements: (3<=n<=15)
10
Learning sequence:
[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]
Use standard params(1) or enter your own(2)?:
2
Number of columns in learning matrix(p) (p>=1 & p<k)
3
Enter max error(e) (0<e<=0.1)
0.001
Enter step learning(alfa) (0<alfa<=0.1 & alfa<=e)
0.007
Enter max number of learning steps(N) (1<=N<=1000000)
150000
Input number of elements to predict: (1<=n<=10)
3
```

Итог:

```
Iteration:  1  Error:  [13.35198684]
Iteration:  2  Error:  [2.95741301]
Iteration:  3  Error:  [2.55583215]
Iteration:  4  Error:  [2.1327443]
Iteration:  5  Error:  [1.74813622]
Iteration:  6  Error:  [1.49096471]
Iteration:  7  Error:  [1.24913184]
Iteration:  8  Error:  [1.04049177]
Iteration:  9  Error:  [0.86282296]
Iteration:  10  Error:  [0.72991891]
Iteration:  10000  Error:  [0.24131451]
Iteration:  20000  Error:  [0.13364419]
Finish learning
Iterations =  23356
Error =  [0.00098038]
[ 8  9 10]
[10.95933814]
[ 9 10 11]
[11.96804891]
[10 11 12]
[12.95773892]
Result sequence:
Result:  [11.]   Expected value:  11   Line error:  [0.]
Result:  [12.]   Expected value:  12   Line error:  [0.]
Result:  [13.]   Expected value:  13   Line error:  [0.]
```

Сеть обучилась за 23356 итераций. Полученные результаты близки к эталонным значениям.

Вывод :

В ходе выполнения лабораторной работы была реализована модель сети Джордана. Было установлено на основе экспериментальных данных, что для различных числовых последовательностей варьируется необходимое количество шагов обучения нейронной сети для достижения максимально-допустимой ошибки. Также было установлено, что в последовательностях сложных для предсказания выход сети отличается от эталонного значения на большую величину, чем в более простых последовательностях.