

## Лабораторна робота 6

### ДОСЛІДЖЕННЯ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ

Мета: *використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися дослідити деякі типи нейронних мереж.*

Хід роботи

#### Завдання 2.1. Ознайомлення з Рекурентними нейронними мережами

					ДУ «Житомирська політехніка».20.121.18.			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Соболевський Д.А.						
Перевір.		Філіпов В.О.						
Керівник								
Н. контр.								
Зав. каф.								
					Літ.	Арк.	Аркушів	
							1	11
					ФІКТ Гр. ІПЗк-20-1			

```

1 import random
2 from data import train_data, test_data
3 import numpy as np
4 from numpy.random import randn
5
6 class RNN:
7     # A many-to-one Vanilla Recurrent Neural Network.
8     def __init__(self, input_size, output_size, hidden_size=64):
9         # Weights
10        self.Whh = randn(hidden_size, hidden_size) / 1000
11        self.Wxh = randn(hidden_size, input_size) / 1000
12        self.Why = randn(output_size, hidden_size) / 1000
13
14        # Biases
15        self.bh = np.zeros((hidden_size, 1))
16        self.by = np.zeros((output_size, 1))
17
18    def forward(self, inputs):
19        '''
20        Perform a forward pass of the RNN using the given inputs.
21        Returns the final output and hidden state.
22        - inputs is an array of one hot vectors with shape (input_size, 1).
23        '''
24        h = np.zeros((self.Whh.shape[0], 1))
25        self.last_inputs = inputs
26        self.last_hs = { 0: h }
27
28        # Perform each step of the RNN
29        for i, x in enumerate(inputs):
30            h = np.tanh(self.Wxh @ x + self.Whh @ h + self.bh)
31            self.last_hs[i + 1] = h
32
33        # Compute the output
34        y = self.Why @ h + self.by
35
36        return y, h
37
38    def backprop(self, d_y, learn_rate=2e-2):
39        '''
40        Perform a backward pass of the RNN.
41        - d_y (dL/dy) has shape (output_size, 1).
42        - learn_rate is a float.
43        '''
44        n = len(self.last_inputs)
45
46        # Calculate dL/dwhy and dL/dby.
47        d_why = d_y @ self.last_hs[n].T
48        d_by = d_y
49
50        # Initialize dL/dwhh, dL/dwxh, and dL/dbh to zero.
51        d_Whh = np.zeros(self.Whh.shape)
52        d_Wxh = np.zeros(self.Wxh.shape)
53        d_bh = np.zeros(self.bh.shape)
54
55        # Calculate dL/dh for the last h.
56        # dL/dh = dL/dy * dy/dh
57        d_h = self.Why.T @ d_y
58
59        # Backpropagate through time.
60        for t in reversed(range(n)):
61            # An intermediate value: dL/dh * (1 - h^2)
62            temp = ((1 - self.last_hs[t + 1] ** 2) * d_h)
63
64            # dL/dbh = dL/dh * (1 - h^2)
65            d_bh += temp
66
67            # dL/dwhh = dL/dh * (1 - h^2) * h_{t-1}
68            d_Whh += temp @ self.last_hs[t].T
69
70            # dL/dwxh = dL/dh * (1 - h^2) * x
71            d_Wxh += temp @ self.last_inputs[t].T
72
73            # Next dL/dh = dL/dh * (1 - h^2) * Whh
74            d_h = self.Whh @ temp
75
76        # Clip to prevent exploding gradients.
77        for d in [d_Wxh, d_Whh, d_why, d_bh, d_by]:
78            np.clip(d, -1, 1, out=d)
79
80        # Update weights and biases using gradient descent.
81        self.Whh -= learn_rate * d_Whh
82        self.Wxh -= learn_rate * d_Wxh
83        self.Why -= learn_rate * d_why
84        self.bh -= learn_rate * d_bh
85        self.by -= learn_rate * d_by
86
87        # Create the vocabulary.
88        vocab = list(set([w for text in train_data.keys() for w in text.split(' ')]))
89        vocab_size = len(vocab)
90
91        print('%d unique words found' % vocab_size)
92
93        # Assign indices to each word.
94        word_to_idx = { w: i for i, w in enumerate(vocab) }
95        idx_to_word = { i: w for i, w in enumerate(vocab) }
96
97        # print(word_to_idx['good'])
98        # print(idx_to_word[0])

```

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

99 def createInputs(text):
100     '''
101     Returns an array of one-hot vectors representing the words in the input text string.
102     - text is a string
103     - Each one-hot vector has shape (vocab_size, 1)
104     '''
105     inputs = []
106
107     for w in text.split(' '):
108         v = np.zeros((vocab_size, 1))
109         v[word_to_idx[w]] = 1
110
111         inputs.append(v)
112
113     return inputs
114 def softmax(xs):
115     # Applies the Softmax Function to the input array.
116     return np.exp(xs) / sum(np.exp(xs))
117
118 # Initialize our RNN!
119 rnn = RNN(vocab_size, 2)
120
121 def processData(data, backprop=True):
122     '''
123     Returns the RNN's loss and accuracy for the given data.
124     - data is a dictionary mapping text to True or False.
125     - backprop determines if the backward phase should be run.
126     '''
127     items = list(data.items())
128
129     random.shuffle(items)
130
131     loss = 0
132     num_correct = 0
133
134     for x, y in items:
135         inputs = createInputs(x)
136         target = int(y)
137
138         # Forward
139         out, _ = rnn.forward(inputs)
140         probs = softmax(out)
141
142         # Calculate loss / accuracy
143         loss -= np.log(probs[target])
144         num_correct += int(np.argmax(probs) == target)
145
146         if backprop:
147             # Build dL/dy
148             d_L_d_y = probs
149             d_L_d_y[target] -= 1
150
151             # Backward
152             rnn.backprop(d_L_d_y)
153
154     return loss / len(data), num_correct / len(data)
155
156 # Training loop
157 for epoch in range(1000):
158     train_loss, train_acc = processData(train_data)
159
160     if epoch % 100 == 99:
161         print('-- Epoch %d' % (epoch + 1))
162         print('Train:\tLoss %.3f | Accuracy: %.3f' % (train_loss, train_acc))
163
164         test_loss, test_acc = processData(test_data, backprop=False)
165
166         print('Test:\tLoss %.3f | Accuracy: %.3f' % (test_loss, test_acc))

```

Рис 1. Лістинг коду файла LR\_6\_task\_1.py

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

18 unique words found
--- Epoch 100
Train: Loss 0.688 | Accuracy: 0.552
Test:  Loss 0.697 | Accuracy: 0.500
--- Epoch 200
Train: Loss 0.666 | Accuracy: 0.655
Test:  Loss 0.728 | Accuracy: 0.450
--- Epoch 300
Train: Loss 0.227 | Accuracy: 0.931
Test:  Loss 0.166 | Accuracy: 0.950
--- Epoch 400
Train: Loss 0.014 | Accuracy: 1.000
Test:  Loss 0.016 | Accuracy: 1.000
--- Epoch 500
Train: Loss 0.006 | Accuracy: 1.000
Test:  Loss 0.007 | Accuracy: 1.000
--- Epoch 600
Train: Loss 0.004 | Accuracy: 1.000
Test:  Loss 0.005 | Accuracy: 1.000
--- Epoch 700
Train: Loss 0.002 | Accuracy: 1.000
Test:  Loss 0.003 | Accuracy: 1.000
--- Epoch 800
Train: Loss 0.002 | Accuracy: 1.000
Test:  Loss 0.002 | Accuracy: 1.000

```

Рис 2. Результат файлу LR\_6\_task\_1.py

Ми спостерігаємо повідомлення на рисунку 1-2 “18 unique words found” це означає, що зміна vocab тепер буде мати перелік всіх слів, які вживаються щонайменше в одному навчальному тексті. Рекурентна нейронна мережа не розрізняє слів – лише числа. Тому у словнику 18 унікальних слів, кожне буде 18-мірним унітарним вектором. І далі відбувається тренування мережі. Виведення кожної соті епохи для відслідковування прогресу

## Завдання 2.2. Дослідження рекурентної нейронної мережі Елмана (Elman Recurrent network (newelm))

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

```

1  import neurolab as nl
2  import numpy as np
3  import pylab as pl
4
5  i1 = np.sin(np.arange(0, 20))
6  i2 = np.sin(np.arange(0, 20)) * 2
7  t1 = np.ones([1, 20])
8  t2 = np.ones([1, 20]) * 2
9  input = np.array([i1, i2, i1, i2]).reshape(20 * 4, 1)
10 target = np.array([t1, t2, t1, t2]).reshape(20 * 4, 1)
11 net = nl.net.newelm([-2, 2]), [10, 1], [nl.trans.TanSig(), nl.trans.PureLin()]
12 net.layers[0].initf = nl.init.InitRand([-0.1, 0.1], 'wb')
13 net.layers[1].initf = nl.init.InitRand([-0.1, 0.1], 'wb')
14
15 net.init()
16
17 # Тренування мережі
18 error = net.train(input, target, epochs=500, show=100, goal=0.01)
19
20 # Запустіть мережу
21 output = net.sim(input)
22
23 # Побудова графіків
24 pl.subplot(211)
25 pl.plot(error)
26 pl.xlabel('Epoch number')
27 pl.ylabel('Train error (default MSE)')
28 pl.subplot(212)
29 pl.plot(target.reshape(80))
30 pl.plot(output.reshape(80))
31 pl.legend(['train target', 'net output'])
32 pl.show()
33

```

Рис 3. Лістинг коду файла LR\_6\_task\_2.py

```

Epoch: 100; Error: 0.24952316020360374;
Epoch: 200; Error: 0.127008826650084;
Epoch: 300; Error: 0.1623061437044942;
Epoch: 400; Error: 0.051613653719392395;
Epoch: 500; Error: 0.05490119280085652;
The maximum number of train epochs is reached

```

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

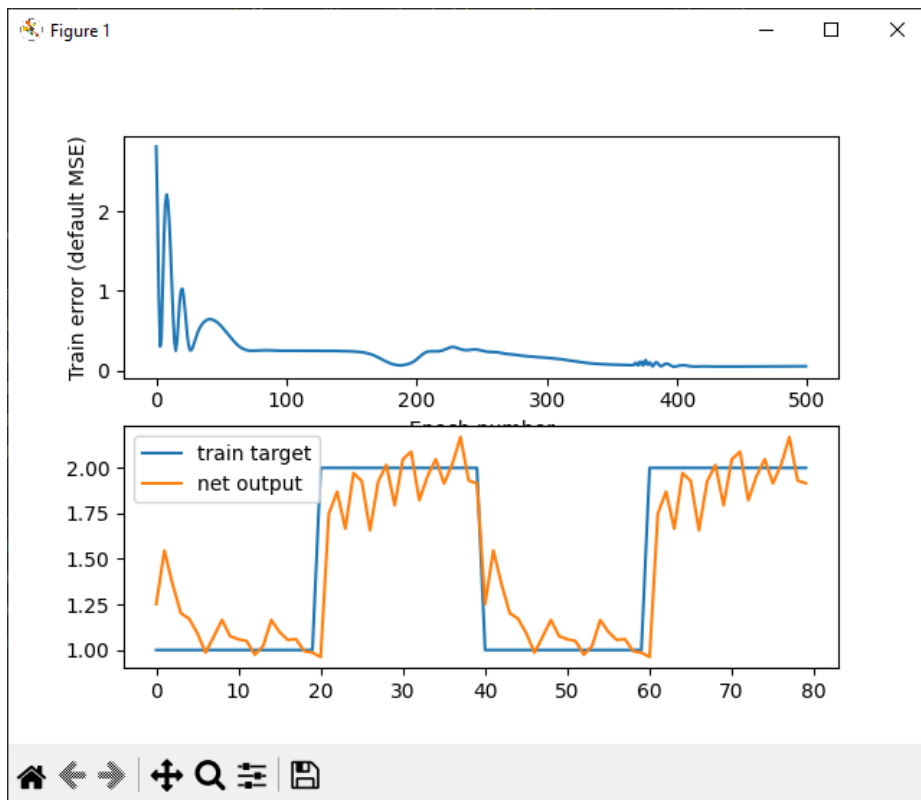


Рис 4. Результат файлу файла LR\_6\_task\_2.py

### Завдання 2.3. Дослідження нейронної мережі Хемінга (Hemming Recurrent network)

```
1 import numpy as np
2 import neurolab as nl
3
4 target = [
5     [-1, 1, -1, -1, 1, -1, -1, 1, -1],
6     [1, 1, 1, 1, -1, 1, 1, -1, 1],
7     [1, -1, 1, 1, 1, 1, 1, -1, 1],
8     [1, 1, 1, 1, -1, -1, 1, -1, -1],
9     [-1, -1, -1, -1, 1, -1, -1, -1, -1]
10 ]
11 input = [
12     [-1, -1, 1, 1, 1, 1, 1, -1, 1],
13     [-1, -1, 1, -1, 1, -1, -1, -1, -1],
14     [-1, -1, -1, -1, 1, -1, -1, 1, -1]
15 ]
16
17 # Створення та тренування нейромережі
18 net = nl.net.newhem(target)
19 output = net.sim(target)
20
21 print("Test on train samples (must be [0, 1, 2, 3, 4])")
22 print(np.argmax(output, axis=0))
23
24 output = net.sim([input[0]])
25
26 print("Outputs on recurrent cycle:")
27 print(np.array(net.layers[1].outs))
28
29 output = net.sim(input)
30
31 print("Outputs on test sample:")
32 print(output)
33
```

Рис 5. Лістинг коду файла LR\_6\_task\_3.py

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PS C:\ztu\штучний інтелект\lab6> python .\LR_6_task_3.py
Test on train samples (must be [0, 1, 2, 3, 4])
[0 1 2 3 4]
Outputs on recurent cycle:
[[0.      0.24    0.48    0.      0.      ]
 [0.      0.144   0.432   0.      0.      ]
 [0.      0.0576  0.4032  0.      0.      ]
 [0.      0.      0.39168  0.      0.      ]]
Outputs on test sample:
[[0.      0.      0.39168  0.      0.      ]
 [0.      0.      0.      0.      0.39168 ]
 [0.07516193 0.      0.      0.      0.07516193]]
PS C:\ztu\штучний інтелект\lab6>

```

Рис 6. Результат файлу файла LR\_6\_task\_3.py

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



## Завдання 2.4. Дослідження рекурентної нейронної мережі Хопфілда Hopfield Recurrent network (newhop)

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

```

1  import numpy as np
2  import neurolab as nl
3
4  target = [
5      [
6          1, 0, 0, 0, 1,
7          1, 1, 0, 0, 1,
8          1, 0, 1, 0, 1,
9          1, 0, 0, 1, 1,
10         1, 0, 0, 0, 1
11     ],
12     [
13         1, 1, 1, 1, 1,
14         1, 0, 0, 0, 0,
15         1, 1, 1, 1, 1,
16         1, 0, 0, 0, 0,
17         1, 1, 1, 1, 1
18     ],
19     [
20         1, 1, 1, 1, 0,
21         1, 0, 0, 0, 1,
22         1, 1, 1, 1, 0,
23         1, 0, 0, 1, 0,
24         1, 0, 0, 0, 1
25     ],
26     [
27         0, 1, 1, 1, 0,
28         1, 0, 0, 0, 1,
29         1, 0, 0, 0, 1,
30         1, 0, 0, 0, 1,
31         0, 1, 1, 1, 0
32     ]
33 ]
34 chars = ['N', 'E', 'R', 'O']
35 target = np.asfarray(target)
36 target[target == 0] = -1
37 net = nl.net.newhop(target)
38 output = net.sim(target)
39
40 print("Test on train samples:")
41
42 for i in range(len(target)):
43     print(chars[i], (output[i] == target[i]).all())
44
45 print("\nTest on defaced M:")
46
47 test = np.asfarray([
48     0, 0, 0, 0, 0,
49     1, 1, 0, 0, 1,
50     1, 1, 1, 1, 1,
51     0, 1, 1, 1, 1,
52     1, 0, 0, 0, 1
53 ])
54 test[test==0] = -1
55 out = net.sim([test])
56
57 print((out[0] == target[1]).all(), 'Sim. steps', len(net.layers[0].outs))

```

Рис 7. Лістинг коду файла LR\_6\_task\_4.py

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PS C:\ztu\штучний інтелект\lab6> python .\LR_6_task_4.py
Test on train samples:
N True
E True
R True
O True

Test on defaced M:
False Sim. steps 3

```

Рис 8. Результат файлу файла LR\_6\_task\_4.py

Як бачимо, навчання пройшло правильно і мережа при невеликій кількості помилок вгадала букви правильно.

**Завдання 2.5. Дослідження рекурентної нейронної мережі Хопфілда для ваших персональних даних**

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1  import numpy as np
2  import neurolab as nl
3
4  target = [
5      [
6          1, 1, 1, 1, 1,
7          1, 0, 0, 0, 0,
8          1, 0, 0, 0, 0,
9          1, 0, 0, 0, 0,
10         1, 1, 1, 1, 1
11     ],
12     [
13         0, 1, 1, 1, 0,
14         0, 1, 0, 1, 0,
15         0, 1, 0, 1, 0,
16         1, 1, 1, 1, 1,
17         1, 0, 0, 0, 1
18     ],
19     [
20         0, 0, 1, 0, 0,
21         0, 0, 1, 0, 0,
22         0, 1, 0, 1, 0,
23         0, 1, 1, 1, 0,
24         0, 1, 0, 1, 0
25     ]
26 ]
27
28 chars = ['C', 'Д', 'А']
29 target = np.asfarray(target)
30 target[target == 0] = -1
31 net = nl.net.newhop(target)
32 output = net.sim(target)
33
34 print("Test on train samples:")
35
36 for i in range(len(target)):
37     print(chars[i], (output[i] == target[i]).all())
38
39 print("\nTest on defaced V:")
40
41 test = np.asfarray([
42     1, 1, 1, 1, 1,
43     1, 0, 0, 0, 1,
44     1, 0, 0, 0, 1,
45     1, 0, 1, 0, 1,
46     1, 0, 0, 0, 0
47 ])
48 test[test==0] = -1
49 out = net.sim([test])
50
51 print ((out[0] == target[0]).all(), 'Sim. steps', len(net.layers[0].outs))

```

Рис 9. Лістинг коду файла LR\_6\_task\_5.py

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PS C:\ztu\штучний інтелект\lab6> python .\LR_6_task_5.py
Test on train samples:
С True
Д True
А True

Test on defaced V:
False Sim. steps 2

```

Рис 10. Результат файлу файла LR\_6\_task\_5.py

Висново: під час виконання лабараторної роботи, використовуючи спеціалізовані бібліотеки та мову програмування Python навчився досліджувати деякі типи нейронних мереж.

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр6	Арк.
		Філіпов В.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		