

ЛАБОРАТОРНА РОБОТА № 6

ДОСЛІДЖЕННЯ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ

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

Хід роботи:

Завдання №1: Дослідження рекурентної нейронної мережі Елмана.

Лістинг програми:

```
import numpy as np
import random

from rnn import RNN
from data import train_data, test_data

# Create the vocabulary.
vocab = list(set([w for text in train_data.keys() for w in text.split(' ')]))
vocab_size = len(vocab)
print('%d unique words found' % vocab_size)

# Assign indices to each word.
word_to_idx = { w: i for i, w in enumerate(vocab) }
idx_to_word = { i: w for i, w in enumerate(vocab) }
# print(word_to_idx['good'])
# print(idx_to_word[0])

def createInputs(text):
    """
    Returns an array of one-hot vectors representing the words in the input text
    string.
    - text is a string
    - Each one-hot vector has shape (vocab_size, 1)
    """
    inputs = []
    for w in text.split(' '):
        v = np.zeros((vocab_size, 1))
        v[word_to_idx[w]] = 1
        inputs.append(v)
    return inputs

def softmax(xs):
    # Applies the Softmax Function to the input array.
    return np.exp(xs) / sum(np.exp(xs))

# Initialize our RNN!
rnn = RNN(vocab_size, 2)

def processData(data, backprop=True):
```

					ДУ «Житомирська політехніка».23.121.9.000 – Лр7			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кормиш Р.І			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Іванов Д.А						Аркушів
Керівник								1
Н. контр.								7
Зав. каф.							ФІКТ Гр.ІПЗ-20-4	

```

'''
Returns the RNN's loss and accuracy for the given data.
- data is a dictionary mapping text to True or False.
- backprop determines if the backward phase should be run.
'''
items = list(data.items())
random.shuffle(items)

loss = 0
num_correct = 0

for x, y in items:
    inputs = createInputs(x)
    target = int(y)

    # Forward
    out, _ = rnn.forward(inputs)
    probs = softmax(out)

    # Calculate loss / accuracy
    loss -= np.log(probs[target])
    num_correct += int(np.argmax(probs) == target)

    if backprop:
        # Build dL/dy
        d_L_d_y = probs
        d_L_d_y[target] -= 1

        # Backward
        rnn.backprop(d_L_d_y)

return loss / len(data), num_correct / len(data)

# Training loop
for epoch in range(1000):
    train_loss, train_acc = processData(train_data)

    if epoch % 100 == 99:
        print('--- Epoch %d' % (epoch + 1))
        print('Train:\tLoss %.3f | Accuracy: %.3f' % (train_loss, train_acc))

        test_loss, test_acc = processData(test_data, backprop=False)
        print('Test:\tLoss %.3f | Accuracy: %.3f' % (test_loss, test_acc))

```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр7	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab6\LR_6_task_1.py"
18 unique words found
--- Epoch 100
Train: Loss 0.688 | Accuracy: 0.534
Test: Loss 0.698 | Accuracy: 0.500
--- Epoch 200
Train: Loss 0.667 | Accuracy: 0.621
Test: Loss 0.716 | Accuracy: 0.450
--- Epoch 300
Train: Loss 0.160 | Accuracy: 0.966
Test: Loss 0.128 | Accuracy: 0.900
--- Epoch 400
Train: Loss 0.008 | Accuracy: 1.000
Test: Loss 0.008 | Accuracy: 1.000
--- Epoch 500
Train: Loss 0.004 | Accuracy: 1.000
Test: Loss 0.008 | Accuracy: 1.000
--- Epoch 600
Train: Loss 0.003 | Accuracy: 1.000
Test: Loss 0.006 | Accuracy: 1.000
--- Epoch 700
Train: Loss 0.002 | Accuracy: 1.000
Test: Loss 0.004 | Accuracy: 1.000
--- Epoch 800
Train: Loss 0.001 | Accuracy: 1.000
Test: Loss 0.003 | Accuracy: 1.000
--- Epoch 900
Train: Loss 0.001 | Accuracy: 1.000
Test: Loss 0.002 | Accuracy: 1.000
--- Epoch 1000
Train: Loss 0.001 | Accuracy: 1.000
Test: Loss 0.001 | Accuracy: 1.000
```

Рис.1 – Результат виконання програми.

Завдання №2: Дослідження рекурентної нейронної мережі Елмана.

Лістинг програми:

```
import neurolab as nl
import numpy as np
import matplotlib.pyplot as plt

# Створення мовлєй сигналу для навчання
i1 = np.sin(np.arange(0, 20))
i2 = np.sin(np.arange(0, 20)) * 2
t1 = np.ones([1, 20])
t2 = np.ones([1, 20]) * 2

input = np.array([i1, i2, i1, i2]).reshape(20 * 4, 1)
target = np.array([t1, t2, t1, t2]).reshape(20 * 4, 1)
```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр7	Арк.
		Іванов Д.А.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Створення мережі з 2 прошарками
net = nl.net.newelm([[ -2, 2]], [10, 1], [nl.trans.TanSig(), nl.trans.PureLin()])

# Ініціалізуйте початкові функції вагів
net.layers[0].initf = nl.init.InitRand([-0.1, 0.1], 'wb')
net.layers[1].initf = nl.init.InitRand([-0.1, 0.1], 'wb')
net.init()

# Тренування мережі
error = net.train(input, target, epochs=500, show=100, goal=0.01)
# Запустіть мережу
output = net.sim(input)

# Побудова графіків

plt.subplot(211)
plt.plot(error)
plt.xlabel('Epoch number')
plt.ylabel('Train error (default MSE)')

plt.subplot(212)
plt.plot(target.reshape(80))
plt.plot(output.reshape(80))
plt.legend(['train target', 'net output'])
plt.show()

"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab6\LR_6_task_2.py"
Epoch: 100; Error: 0.24706385565235242;
Epoch: 200; Error: 0.07208981944058572;
Epoch: 300; Error: 0.07674216484023202;
Epoch: 400; Error: 0.038761089629510165;
Epoch: 500; Error: 0.037827084646633695;
The maximum number of train epochs is reached
```

Рис.2 – Результат виконання програми.

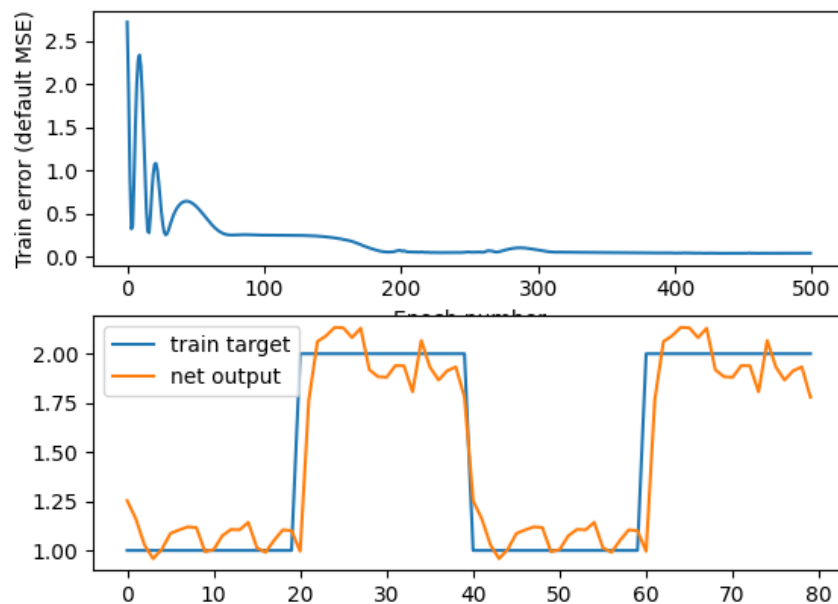


Рис.3 – Результат виконання програми.

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр7	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

Завдання №3: Дослідження нейронної мережі Хемінга.

Лістинг програми:

```
import numpy as np
import neurolab as nl

target = [[-1, 1, -1, -1, 1, -1, -1, 1, -1],
          [1, 1, 1, 1, -1, 1, 1, -1, 1],
          [1, -1, 1, 1, 1, 1, 1, -1, 1],
          [1, 1, 1, 1, -1, -1, 1, -1, -1],
          [-1, -1, -1, -1, 1, -1, -1, -1, -1]]

input = [[-1, -1, 1, 1, 1, 1, 1, -1, 1],
         [-1, -1, 1, -1, 1, -1, -1, -1, -1],
         [-1, -1, -1, -1, 1, -1, -1, 1, -1]]

# Створення та тренування нейромережі
net = nl.net.newhem(target)

output = net.sim(target)
print("Test on train samples (must be [0, 1, 2, 3, 4])")
print(np.argmax(output, axis=0))

output = net.sim([input[0]])
print("Outputs on recurent cycle:")
print(np.array(net.layers[1].outs))

output = net.sim(input)
print("Outputs on test sample:")
print(output)
```

```
"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab6\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]]
```

Рис.4 – Результат виконання програми.

Завдання №4: Дослідження рекурентної нейронної мережі Хопфілда.

Лістинг програми:

```
import numpy as np
import neurolab as nl

target = [[1, 0, 0, 0, 1,
          1, 1, 0, 0, 1,
```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр7	Арк.
		Іванов Д.А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1, 0, 1, 0, 1,
1, 0, 0, 1, 1,
1, 0, 0, 0, 1],
[1, 1, 1, 1, 1,
1, 0, 0, 0, 0,
1, 1, 1, 1, 1,
1, 0, 0, 0, 0,
1, 1, 1, 1, 1],
[1, 1, 1, 1, 0,
1, 0, 0, 0, 1,
1, 1, 1, 1, 0,
1, 0, 0, 1, 0,
1, 0, 0, 0, 1],
[0, 1, 1, 1, 0,
1, 0, 0, 0, 1,
1, 0, 0, 0, 1,
1, 0, 0, 0, 1,
0, 1, 1, 1, 0]]

chars = ['N', 'E', 'R', 'O']
target = np.asfarray(target)
target[target == 0] = -1

# Create and train network
net = nl.net.newhop(target)

output = net.sim(target)
print("Test on train samples:")
for i in range(len(target)):
    print(chars[i], (output[i] == target[i]).all())

print("\nTest on defaced N:")
test = np.asfarray([0, 0, 0, 0, 0,
                    1, 1, 0, 0, 1,
                    1, 1, 0, 0, 1,
                    1, 0, 1, 1, 1,
                    0, 0, 0, 1, 1])

test[test == 0] = -1
out = net.sim([test])
print((out[0] == target[0]).all(), 'Sim. steps', len(net.layers[0].outs))

"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab6\LR_6_task_4.py"
Test on train samples:
N True
E True
R True
O True

Test on defaced N:
True Sim. steps 2

```

Рис.5 – Результат виконання програми.

Завдання №5: Дослідження рекурентної нейронної мережі Хопфілда.

Лістинг програми:

```

import numpy as np
import neurolab as nl

target = [[1, 0, 0, 1, 0,

```

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр7	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

1, 0, 1, 0, 0,
1, 1, 0, 0, 0,
1, 0, 1, 0, 0,
1, 0, 0, 1, 0],
[1, 1, 1, 1, 1,
1, 0, 0, 0, 1,
1, 0, 0, 0, 1,
1, 0, 0, 0, 1,
1, 1, 1, 1, 1],
[0, 0, 1, 0, 0,
0, 0, 1, 0, 0,
0, 0, 1, 0, 0,
0, 0, 1, 0, 0,
0, 0, 1, 0, 0,
0, 0, 1, 0, 0]]

chars = ['K', 'O', 'I']
target = np.asfarray(target)
target[target == 0] = -1

# Create and train network
net = nl.net.newhop(target)

output = net.sim(target)
print("Test on train samples:")
for i in range(len(target)):
    print(chars[i], (output[i] == target[i]).all())

print("\nTest on defaced I:")
test = np.asfarray([0, 0, 1, 0, 0,
                    0, 0, 1, 0, 0,
                    0, 0, 1, 0, 0,
                    0, 0, 1, 0, 0,
                    0, 1, 0, 0, 0])

test[test == 0] = -1
out = net.sim([test])
print((out[0] == target[0]).all(), 'Sim. steps', len(net.layers[0].outs))

"D:\it\Forth Course\AI\venv\Scripts\python.exe" "D:\it\Forth Course\AI\Lab6\LR_6_task_5.py"
Test on train samples:
K True
O True
I True

Test on defaced I:
False Sim. steps 1

```

Рис.7 – Результат виконання програми.

Посилання на GitHub: <https://github.com/Raimhal1/AI>

Висновки: в ході лабораторної роботи ми використовуючи спеціалізовані бібліотеки та мову програмування Python навчилися дослідити деякі типи нейронних мереж.

		Кормиш Р.І			ДУ «Житомирська політехніка».23.121.9.000 – Лр7	Арк.
		Іванов Д.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		