Softcomputing
Neural network
Report

Arkadiusz Szydełko 184626
Tomasz Kuchne 184674

May 1, 2015

1 Presentation of the problem

The idea behind the project is to translate written by hand text to a machine text. There are a few problems to solve. The first one is to convert graphical image to number representation of each character. This part is already done, there are matrix representations of each character. The rest of the process has to be implemented.

The main problem of this project is to convert given matrices to real characters. Given matrices could not perfectly reflect correct representations of coded characters. There can be several problems, e.g. matrices can contain noises, values in different positions, values could be different or the matrices can even be different sizes.

Figures 1, 2 and 3 shows more precariously how the data is represented and what are the possible problems.

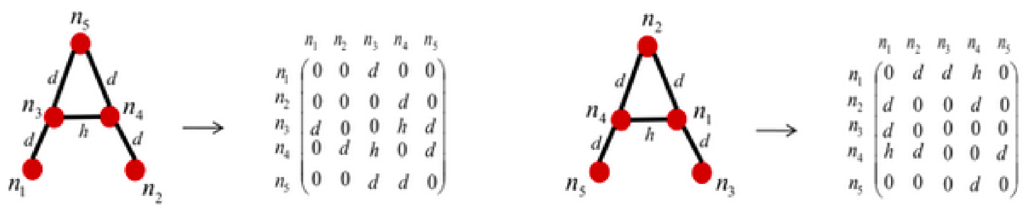


Figure 1: Different position of nodes.

On the figure 1, it can be observed that, when convention of node numeration will change, the matrix will be different, even if it contains the same number of values of each type.

Each letter has different number of nodes. The nodes are representing a critical points of the letter. E.g. letter A has 5 nodes, which connected in a correct way construct letter A. The values itself are representing connection type between letter's nodes. Overall there are 4 correct type of connections, but it happens that matrices will contain other not specified types caused by errors. Including incorrect types the number of types which appear in the data is 8.

Figures 2 and 3 represents other types of errors in matrices, with noise, different values and different matrix size.

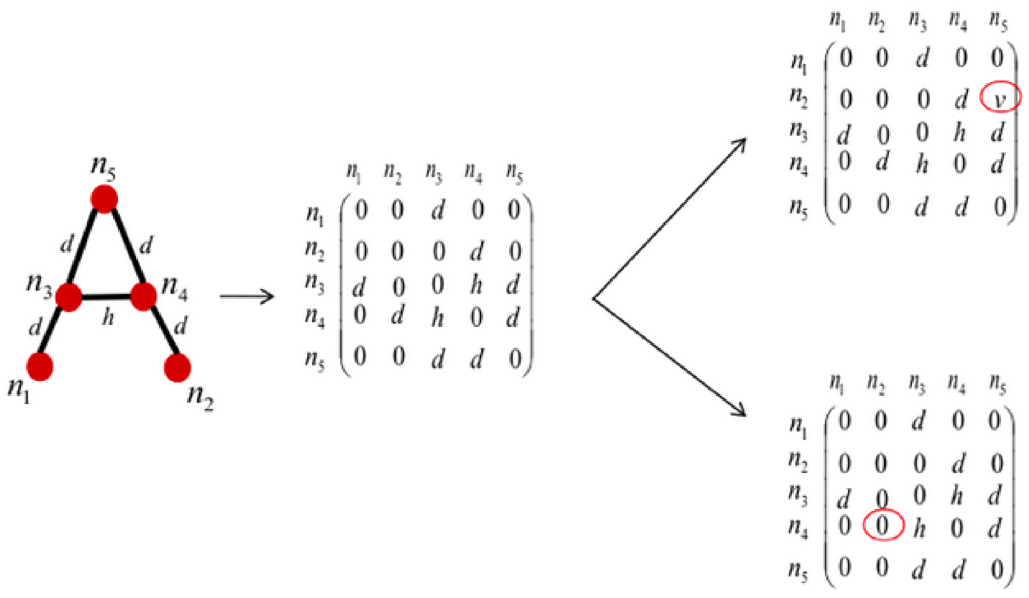


Figure 2: Noise example.

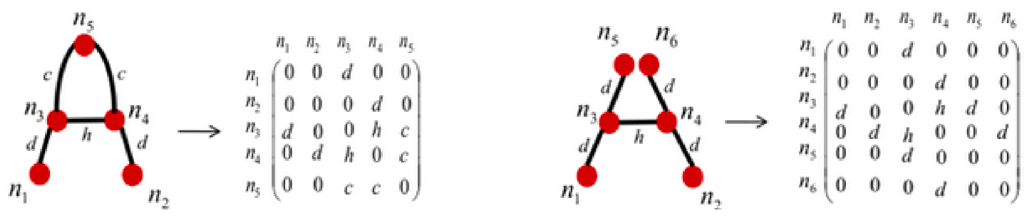


Figure 3: Different values and sizes.

2 Data preparation

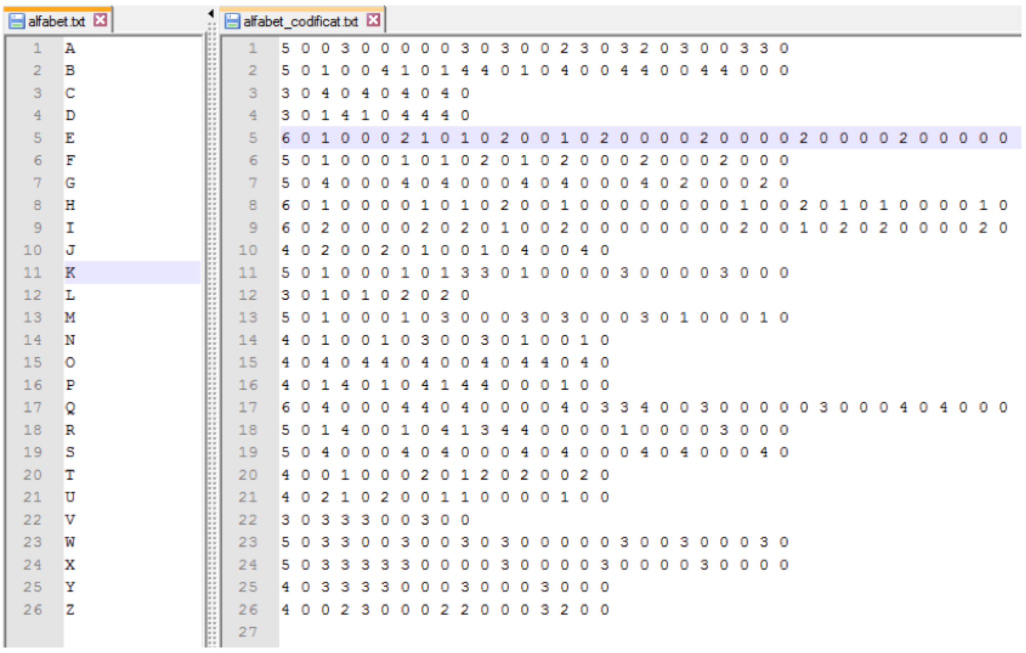
The data to work on is provided in a text files where each line represent different letter and can be transformed into matrix. Data which represent alphabet is shown in Figure 4. The first number in the line determinate how many columns given matrix contain, rest of numbers in the line are values of matrices.

Each letter is a matrix of different size and is not in a binary representation, which is not suitable for neural network usage. Because of that the data need to be transformed into alternative representation.

The biggest matrix which appear in the test cases has size $10x10$, based on this assumption all other matrices were resized to fit this size. Matrices were resized by adding 0 on the right and bottom sides. After that values needed to be transformed into binary representation. To accomplish that each position was extended (in width of the matrix) to have 4 digits. As mentioned in section 1, the highest value that appear in test cases is 8 and to represent 8 in binary code, 4 positions are needed.

After all transformations all matrices has the same size, contains only binary values and their size is $40x10$. Which is a result of calculations 1. E.g. representation of the letter A is illustrated in figure 5.

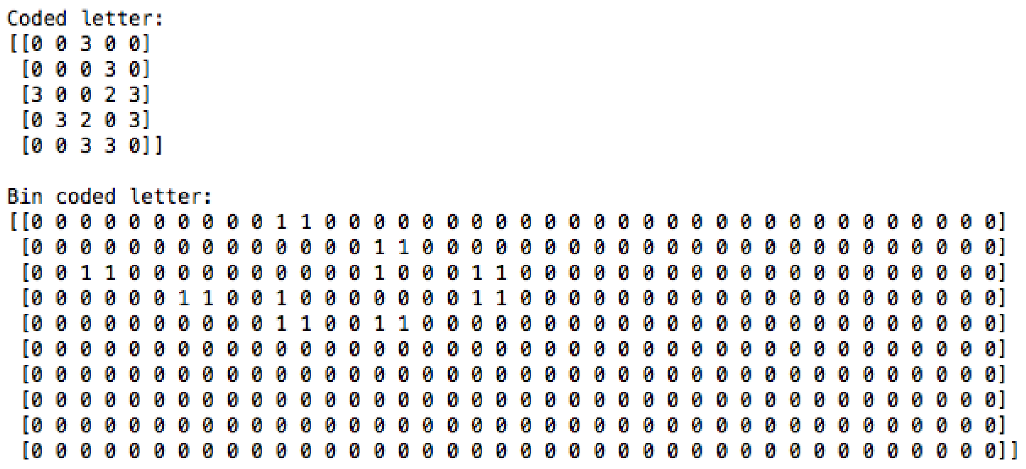
$$\text{(10 columns} * \text{4 bit) } x \text{ 10 rows} = \text{40 columns x 10 rows} \quad (1)$$



```
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
```

```
5 0 0 3 0 0 0 0 0 3 0 3 0 0 2 3 0 3 2 0 3 0 0 3 3 0
5 0 1 0 0 4 1 0 1 4 4 0 1 0 4 0 0 4 4 0 0 4 4 0 0 0
3 0 4 0 4 0 4 0 4 0
3 0 1 4 1 0 4 4 4 0
6 0 1 0 0 0 2 1 0 1 0 2 0 0 1 0 2 0 0 0 0 2 0 0 0 0 2 0 0 0 0 2 0 0 0 0 0
5 0 1 0 0 0 1 0 1 0 2 0 1 0 2 0 0 0 2 0 0 0 2 0 0 0
5 0 4 0 0 0 4 0 4 0 0 0 4 0 4 0 0 0 4 0 2 0 0 0 2 0
6 0 1 0 0 0 0 1 0 1 0 2 0 0 1 0 0 0 0 0 0 0 1 0 0 2 0 1 0 1 0 0 0 0 1 0
6 0 2 0 0 0 0 2 0 2 0 1 0 0 2 0 0 0 0 0 0 0 0 2 0 0 1 0 2 0 2 0 0 0 0 2 0
4 0 2 0 0 2 0 1 0 0 1 0 4 0 0 4 0
5 0 1 0 0 0 1 0 1 3 3 0 1 0 0 0 0 3 0 0 0 0 3 0 0 0
3 0 1 0 1 0 2 0 2 0
5 0 1 0 0 0 1 0 3 0 0 0 3 0 3 0 0 0 3 0 1 0 0 0 1 0
4 0 1 0 0 1 0 3 0 0 3 0 1 0 0 1 0
4 0 4 0 4 4 0 4 0 0 4 0 4 4 0 4 0
4 0 1 4 0 1 0 4 1 4 4 0 0 0 1 0 0
6 0 4 0 0 0 4 4 0 4 0 0 0 0 4 0 3 3 4 0 0 3 0 0 0 0 0 3 0 0 0 4 0 4 0 0 0
5 0 1 4 0 0 1 0 4 1 3 4 4 0 0 0 0 1 0 0 0 0 3 0 0 0
5 0 4 0 0 0 4 0 4 0 0 0 4 0 4 0 0 0 4 0 4 0 0 0 4 0
4 0 0 1 0 0 0 2 0 1 2 0 2 0 0 2 0
4 0 2 1 0 2 0 0 1 1 0 0 0 0 1 0 0
3 0 3 3 3 0 0 3 0 0
5 0 3 3 0 0 3 0 0 3 0 3 0 0 0 0 0 3 0 0 3 0 0 0 3 0
5 0 3 3 3 3 3 0 0 0 0 3 0 0 0 0 3 0 0 0 0 3 0 0 0 0
4 0 3 3 3 3 0 0 0 3 0 0 0 3 0 0 0
4 0 0 2 3 0 0 0 2 2 0 0 0 3 2 0 0
```

Figure 4: Alphabet representation.



```
Coded letter:
[[0 0 3 0 0]
 [0 0 0 3 0]
 [3 0 0 2 3]
 [0 3 2 0 3]
 [0 0 3 3 0]]

Bin coded letter:
[[0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

Figure 5: Representation of coded letter A.

3 Neural network

Network used in the project is a multilayer perceptron network with recurrent connections in the hidden layer. It is characteristic of having multiple layers of nodes. Two obligatory layers, input and output layers and hidden layers (one or more). The recurrent part in addition to a traditional connections in multilayer perceptron, has also connections between each node in the hidden layer, including node connections to themselves. This approach is useful to handle sequential data. Because it can use their internal memory to process sequences it is an ideal solution for handwriting recognition. Unfortunately the data provided to this project is not suitable to fully make use of this kind of network, but it was discussed on project meeting to stay with that solution.

3.1 Structure

The network created for the project has 400 nodes in input layer, one hidden layer where various number of nodes were tested and 26 nodes in output layer. The size of input layer is caused by sizes of data matrices, which are of size $40x10$. After translating that matrix into a vector, which is required as an input structure, it will be with a length of 400. The size of hidden layer was tested for various numbers of nodes, but the best results were achieved with 200 nodes in the hidden layer. The size of output layer is equal to 26, because the alphabet contains 26 letter and implementation of network in used library is working in a way that the output has only one value, which is equal to 1, the rest are 0.

Total number of connections in the network is equal to 125 200. With the assumption that there are 200 nodes in hidden layer. The value was calculated by equation 2.

$$N_w = N_{il} * N_{hl} + N_{hl} * N_{hl} + N_{hl} * N_{ol} \tag{2}$$

N_w	- Number of connections
N_{il}	- Number of nodes in input layer
N_{hl}	- Number of nodes in hidden layer
N_{ol}	- Number of nodes in output layer

3.2 Training

The network was trained with backpropagation method. Number of training episodes was chosen based on total error value. Each time learning was started, total error value at the beginning was decreasing to some point and then when the training was continued, the value was slowly growing with each next episode. The final number of training iteration was chosen on a point where total error value was the smallest possible, which was different for various network configurations.

4 Tests

There is a set of six tests cases. Each of them contains different types of errors.

T_1	- Test case does not contain any errors.
T_2	- Test case contains mixed rows and columns.
T_3	- Test case contains mixed rows and columns.
T_4	- Test case contains noise in matrices.
T_5	- Test case contains noise in matrices and changed values in correct positions.
T_6	- Test case contains noise in matrices, mixed rows and columns, changed values and changed matrix sizes.

The most interesting results are presented in the table 1, for others configuration parameters the results ware not satisfactory. Following range of parameters were tasted:

Learning rate	- Values: 0,05 , 0,1 and 0,2.
Momentum	- Values: 0,1 and 0,2.
Nodes in hidden layer	- Values: 50, 100 and 200.

No	Learning Rate	Momentum	Nodes in hidden layer	Training iterations	T_1	T_2	T_3	T_4	T_5	T_6
1	0,05	0,1	50	200	94%	68%	46%	96%	42%	88%
2	0,05	0,1	200	180	97%	65%	53%	98%	43%	88%
3	0,05	0,2	50	100	100%	72%	51%	100%	47%	91%
4	0,05	0,2	200	120	100%	70%	61%	99%	48%	91%
5	0,1	0,1	50	40	97%	68%	46%	97%	43%	88%
6	0,1	0,2	100	40	100%	70%	57%	100%	46%	91%

Table 1: Recognition results.

From performed tests overall the best configuration is the number 4 from the table 1. It achieved the highest percentage of correct recognition in most test cases.

The results shows in what situations the network is doing well or not. Clearly the easiest case for the network is to recognize letters with noise, the results for this test case were almost 100% across different configurations.

The network has problems when letters are represented correctly, but in different order of nodes, test cases no. 2 and 3. The results vary from 46% to 72% of correct recognition. But it is understandable when taking into assumption the way that neural network works and the way of data binary representation. After such a changes matrix can clearly start to look similar to different letter.

The worst cases for the network to deal with are noises connected with changed values, illustrated in figure 2 and 3. The reason is probably similar to problems with test cases no. 2 and 3, but results are much worst. In any configuration even 50% of correct recognition was not achieved.

The last test case which contains all kind of errors went well with average of 90% of correct recognition. Wrong recognition was mostly connected with errors of different order of nodes and changed values.

5 Implementation

Project was implemented in Python 2.7, with additional libraries to handle network structure and learning algorithms. Following libraries were used:

- PyBrain - library which provide tool to manage different types of machine learning procedures and structures. It contains modules to handle neural networks.
- NumPy - package for scientific computing with Python, with native functions written in C/C++ for faster computations and more efficient number date representations. It also provide functions for various matrices operations.
- SciPy - module to support mathematics operation in Python.

Project was implemented in a way for easy possibility to change different network parameters in order to test various learning and network options. Project also contain analytics module which create a special files with analysis of each computed letter. The analysis contains given letter, expected letter and obtained letter in all possible representations (as letter, as bin matrix and as original matrix) to easily investigate network behaviors.

6 Conclusions

With right tools basic neural networks can be easily implemented. Neural networks are great for problems connected with data that contains noise or are deformed. Unfortunately it is not good solution for modified data sets, where some values are different then they should be (problems illustrated in figure 3).