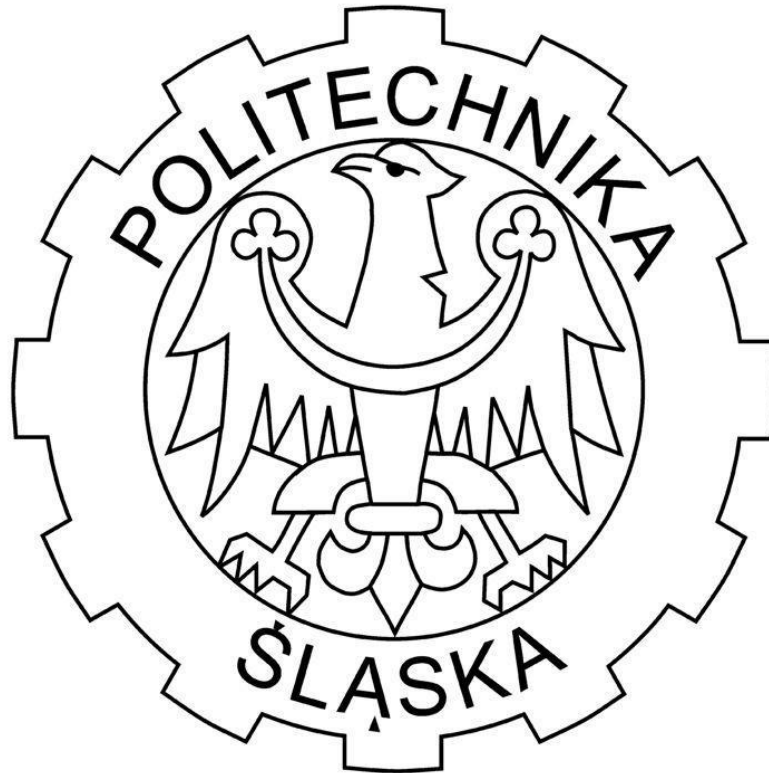


Biologically Inspired Artificial Intelligence

Project report



Jacek Ganszczyk,
Piotr Kałucki,
SSI,
GKiO2

1.Introduction

Our target was to write a program using Neural Network able to recognize shapes that are represented as black and white pictures 28px x 28px.

2.Assumptions

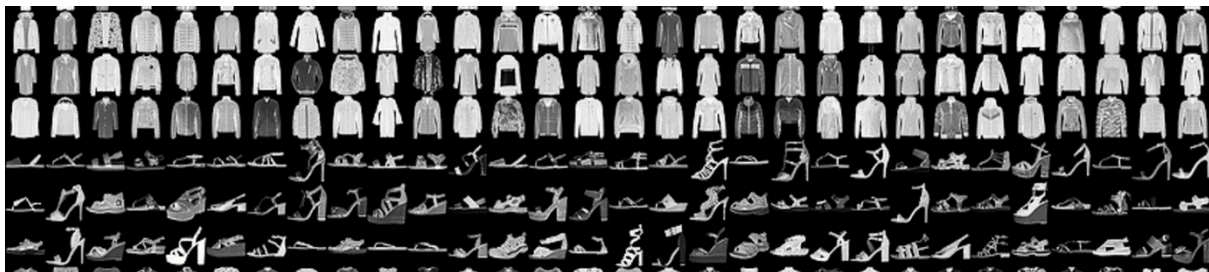
- Program was written using C++ language
- Program should recognize shapes represented as 28x28 table
- For testing we used 2 datasets:
MNIST – handwritten numbers 0-9
FashionMist – zalando clothes categories

3.Datasets

Datasets contains 60,000 samples evenly speeded across.

Mnist have numbers from 0-9[labels]

And the FashionMnist:



- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

3.Program


Program at the beginning use random function to fulfill weights. All data in program are stored in global variables, weights and neurons values are stored in tables.

During learning process neural network use backpropagation to modify weights.

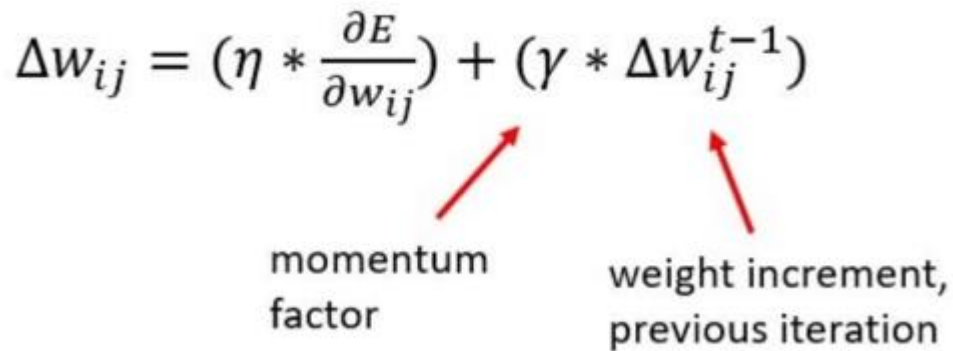
So first we feed forward our network until we get output. We compare our output with label. Then we can calculate that gradient which is needed for backpropagate change of weights.

Then we take learning rate and gradient multiply by each other and adding to this momentum multiplied by previous delta.

That results in delta of weight change applied to the model.

$$\Delta w_{ij} = \left(\eta * \frac{\partial E}{\partial w_{ij}} \right)$$


weight increment learning rate weight gradient

$$\Delta w_{ij} = \left(\eta * \frac{\partial E}{\partial w_{ij}} \right) + (\gamma * \Delta w_{ij}^{t-1})$$


momentum factor

weight increment, previous iteration

Error is a sum of square differences between target value and prediction. That error compared to epsilon is used for checking how well the learning process is going. When error is lower than epsilon learning for sample is ended. Otherwise learning is ended when number of epochs is reached

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

Program while learning generates report which is saved to file. Looking into this report shows us how it is going.

```
(Sample number870) iteration number: 273, Error = 0.000997581
(Sample number871) iteration number: 512, Error = 0.00135195
(Sample number872) iteration number: 1, Error = 0.000740836
(Sample number873) iteration number: 380, Error = 0.000998412
(Sample number874) iteration number: 277, Error = 0.000997419
(Sample number875) iteration number: 343, Error = 0.00099814
(Sample number876) iteration number: 8, Error = 0.000998168
```

Thanks to that data we were able to left computer for long time and after process was done see how well it gone. Any learning issues would showed up in that report.

Also we generate files with testing results:

```
(Sample number9992)    TRUE           , answer: 8, correct answer: 8. Error was 0.0292595
(Sample number9993)    TRUE           , answer: 9, correct answer: 9. Error was 0.275095
(Sample number9994)    TRUE           , answer: 0, correct answer: 0. Error was 0.120354
(Sample number9995)    TRUE           , answer: 1, correct answer: 1. Error was 0.00140042
(Sample number9996)    TRUE           , answer: 2, correct answer: 2. Error was 0.00238889
(Sample number9997)    TRUE           , answer: 3, correct answer: 3. Error was 0.00348724
(Sample number9998)    TRUE           , answer: 4, correct answer: 4. Error was 0.0973766
(Sample number9999)    FALSE          , answer: 8, correct answer: 5. Error was 0.397794
(Sample number10000)   TRUE           , answer: 6, correct answer: 6. Error was 0.000766179
Number of correct samples: 8968 / 10000
Accuracy: 89.68
Label: 0 Occurance980 Postive predictions: 931 95%
Label: 1 Occurance1135 Postive predictions: 1106 97.4449%
Label: 2 Occurance1032 Postive predictions: 938 90.8915%
Label: 3 Occurance1010 Postive predictions: 911 90.198%
Label: 4 Occurance982 Postive predictions: 919 93.5845%
Label: 5 Occurance892 Postive predictions: 645 72.3094%
Label: 6 Occurance958 Postive predictions: 909 94.8852%
Label: 7 Occurance1028 Postive predictions: 906 88.1323%
Label: 8 Occurance974 Postive predictions: 841 86.345%
Label: 9 Occurance1009 Postive predictions: 862 85.4311%
Epochs: 512 Learning rate: 0.01 Epsilon: 0.01 Hidden Neuron count: 128
Samples learned: 6000
```

That data provides us information about tests, they were crucial to make some comparison. We were getting feedback from network using these reports.

4.Models

We learned our network many times to see how it behave when we change parameters like epsilon and learning rate.

We provided into our repositorium 3 model learned for whole 60k samples in both 1hiddenLayer and 2hiddenLater networks. Learning such network take a while. So for more tests we decide to learn network from 6k samples, that was enough to see changing in accuracy regarding input parameters.

5. Test 1 layer

First test showed best result that we achieved during this project. Over 94% in 1 layer. It is on MNIST numbers dataset, all numbers were 90%+.

Label	% of accuracy
0	98.7755%
1	99.1189%
2	93.2171%
3	95.6436%
4	90.0204%
5	92.4888%
6	96.1378%
7	91.2451%
8	90.7598%
9	95.1437%
Total:	94.33%

samples	60k
testing samples	10k
epochs	512
learning rate	0,001
momentum	0,9
epsilon	0,001
hidden neurons	128

Second test was done on Fashion data set where we achieved maximum of 73% for long run at whole dataset.

Label	% of accuracy
0 T-shirt/top	59.4%
1 Trouser	91.2%
2 Pullover	74.5%
3 Dress	92%
4 Coat	25.2%
5 Sandal	90.3%
6 Shirt	54.7%
7 Sneaker	80.4%
8 Bag	93.9%
9 Ankle boot	68.4%
Total:	73%

samples	60k
testing samples	10k
epochs	512
learning rate	0,001
momentum	0,9
epsilon	0,001
hidden neurons	128

5. Second Hidden layer

We were supposed to add second hidden layer to our network, we changed a bit our program to achieve this. With second layer training process get more time consuming, so for comparison we decided not to change any variables from previous long run test. We added second hidden layer with the same amount of neurons.

It was a bit surprising that for numbers dataset we didn't get better results, it was still around 94%

Label	% of accuracy
0	99.0816%
1	98.4141%
2	91.376%
3	97.1287%
4	94.0937%
5	87.7803%
6	97.3904%
7	93.9689%
8	86.8583%
9	94.45%
Total:	94.17%

samples	60k
testing samples	10k
epochs	512
learning rate	0,001
momentum	0,9
epsilon	0,001
hidden neurons	128

And even worse for fashion dataset:

Label	% of accuracy
0 T-shirt/top	19%
1 Trouser	91.7%
2 Pullover	5%
3 Dress	72%
4 Coat	36.1%
5 Sandal	88.4%
6 Shirt	90.2%
7 Sneaker	97.8%
8 Bag	93.4%
9 Ankle boot	60.2%
Total:	65.38%

samples	60k
testing samples	10k
epochs	512
learning rate	0,001
momentum	0,9
epsilon	0,001
hidden neurons	128

We can see that network have the same problems with recognizing “4” and “8” in both test. And in second dataset network had more problems with recognizing Coats and Ankle boots, what was interesting adding second layer changed dramatically accuracy for recognizing shirts.

For sure we could tweak our 2 layer network to get better results, but for limited time we decided to provide more tests on 1 layer network.

6.Tests with different parameters

Our reference test was done on 6k sample for learn and 0.001 learning rate/epsilon. It gave us really nice result overall around 90% for only 6k samples:

Label	% of accuracy
0	95%
1	97.4449%
2	90.8915%
3	90.198%
4	93.5845%
5	72.3094%
6	94.8852%
7	88.1323%
8	86.345%
9	85.4311%
Total:	89.68%

samples	6k
testing samples	10k
epochs	512
learning rate	0,001
momentum	0,9
epsilon	0,001
hidden neurons	128

Then we played with learning rate, we changed it from 0.001 to 0.01, it resulted in better accuracy around +1%

Label	% of accuracy
0	95%
1	98.326%
2	90.8915%
3	92.5743%
4	94.7047%
5	81.7265%
6	95.7203%
7	88.2296%
8	83.6756%
9	81.665%
Total:	90.42%

samples	6k
testing samples	10k
epochs	512
learning rate	0,01
momentum	0,9
epsilon	0,001
hidden neurons	128

Changing it more to 0.1 resulted in decreased accuracy.

Label	% of accuracy
0	81.0204%
1	40.9692%
2	81.686%
3	91.0891%
4	88.2892%
5	54.7085%
6	94.7808%
7	86.4786%
8	79.4661%
9	86.9177%
Total:	78.25%

samples	6k
testing samples	10k
epochs	512
learning rate	0,1
momentum	0,9
epsilon	0,001
hidden neurons	128

After this tests we go for playing with epsilon value, and we did the same steps 0.001->0.01->0.1.

Label	% of accuracy
0	93.1633%
1	97.6211%
2	89.6318%
3	90.7921%
4	91.5479%
5	75%
6	96.8685%
7	87.9377%
8	81.3142%
9	85.1338%
Total:	89.14%

samples	6k
testing samples	10k
epochs	512
learning rate	0,001
momentum	0,9
epsilon	0,01
hidden neurons	128

Label	% of accuracy
0	93.5714%
1	96.5639%
2	87.0155%
3	90.7921%
4	89.4094%
5	79.5964%
6	95.1983%
7	86.965%
8	78.4394%
9	87.4133%
Total:	88.68%

samples	6k
testing samples	10k
epochs	512
learning rate	0,001
momentum	0,9
epsilon	0,1
hidden neurons	128

That tests showed us that epsilon at level 0.001 was best option.

We decided to make one more test combining learning rate and epsilon change.

Label	% of accuracy
0	95%
1	97.4449%
2	90.8915%
3	90.198%
4	93.5845%
5	72.3094%
6	94.8852%
7	88.1323%
8	86.345%
9	85.4311%
Total:	89.68%

samples	6k
testing samples	10k
epochs	512
learning rate	0,01
momentum	0,9
epsilon	0,01
hidden neurons	128

And one more time we realized that our reference sample was the best. For that kind of data both learning rate and epsilon should be at the same value of 0.001. Going lower make really low performance in learning process and not provide better results.

7.Summary

For us project was really interesting because both of us didn't have a lot of knowledge before in that subject. We learned a lot about mathematic that is background of neural networks. This implementation is really basic one, but this is good starting point of understanding problems that are connected with it. We achieved accuracy rates above 90% what we could call a success.

Repo: https://github.com/gancu/nn_biai