

Document Analysis

Exercise : Handwritten Digits Recognition with Artificial Neural Networks

Kai Chen, Mathias Seuret, and Rolf Ingold

Objective

- Experiment with Neural Networks (NNs) on a real problem to understand how NNs work. Study the following aspects:
 - Impact of topology of NNs
 - Impact of parameters, i.e., learning rate, number of epochs
 - Impact of training set size
 - Overfitting

Problem statement

- Using NNs to recognize handwritten digits (0-9)



Data

- MNIST dataset
 - Gray scale PNG images
 - Img-lbl*digit*-number-num.png, e.g., img-lbl**0**-1-num.png
 - Size : 28x28 pixels
 - Pixel intensity value : 0-255
 - Number of images per datasets
 - Training : 31920
 - Test: 10000
 - Validation : 1000

<http://yann.lecun.com/exdb/mnist/>

Tool

- Java application

```
ck@ck-unifr: /media/ck/volume1/unifr/cours/DAR/scripts
File Edit View Search Terminal Help
ck@ck-unifr:/media/ck/volume1/unifr/cours/DAR/scripts$ java -jar nn.jar
Usage: java -jar nn.jar [options] input_training_file input_test_file output_training_log_file output_results_file
options:
-a activation_function:      Activation function: SIGMOID, SOFTSIGN, TANH
-f feature_size:            Feature size (number of input).
-n number_neurons:          Neurons on hidden layer, e.g., 100. If you have several hidden layers, then number of neurons on each layers is seperated by comma, e.g., 100,50
-o output_size:             Number of classes.
-l learning_rate:           Learning rate. By default 0.001.
-e number_epochs:           Number of training epochs.
ck@ck-unifr:/media/ck/volume1/unifr/cours/DAR/scripts$
```

Example:

```
java -jar -Xmx512m nn.jar -a SIGMOID -f 784 -n 100 -o 10 -l 0.001 -e 10
mnist.train.txt mnist.test.txt mnist.train.output.txt mnist.test.output.txt
```

Tool

- Input file format
 - class_label, feature₁, feature₂, ...
 - Features are float values

9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Class

Features (here: raw pixel values)

Tool

- Output file format
 - Train

```
0.17970781  
0.17752841  
0.16313086  
0.14881098  
0.13672492  
0.12724894  
0.120099396  
0.112911336  
0.10735046  
0.10313332|
```

Training error rate per epoch

- Test

```
result: 8,8  
result: 5,2  
result: 3,3  
result: 4,4  
result: 9,9  
result: 6,6  
result: 2,2  
result: 4,4  
result: 6,6  
result: 5,2  
result: 1,1  
result: 3,3  
result: 0,0  
accuracy: 0.8186
```

Results: class label, predict label

Average accuracy

Suggested parameters

- Number of neurons for one layer NNs
 - 5, 10, 50, 100, 200, 500, 784, 1000, ...
- Number of epochs
 - 5, 10, 50, 100, ...
- Learning rate
 - ..., 0.0001, 0.001, 0.01, 0.1, ...
- Training samples
 - 20, 100, 1000, 2000, 5000, 10000, ...

Tasks and Schedule

- Task 1 (14th May)
 - Generate feature files containing class label with pixel values
 - Evaluate one layer NNs
 - Use different parameter values and observe the performance
 - #neurons, learning rate, #epochs
 - Optional : Evaluate NNs with more than one layer
- Task 2 (28th May)
 - Replace pixel values by high-level set of features
 - e.g., scale down images and extract pixel intensity values
 - Use different number of training samples

Hints

- Normalize the features from -1 to 1
- Before doing a test with many epochs, try it with only 3-4 to see if it runs correctly
- Write scripts to run bunch of test
- The initial weights are random
 - Run twice the same test, you will get two different results
 - Very small differences of accuracy might not be relevant
 - In such cases, take the average of several runs
- Keep track of the experiments you did
 - So that you don't lose time doing them twice

Deadline

- May 28, 15:00
- Hand in a ZIP file “HansMuster_JaneDoe_JohnDoe.zip” via ILIAS that contains:
 - Your source code
 - Report with descriptions and figures (PDF)