Programming Assignment 1

NN & CNN

CSE303: Introduction to Deep Learning

Objective

1. **3-layer Neural Network for Classification**
    1. without the deep learning framework (only python)
    2. using a deep learning framework (e.g. pytorch, tensorflow)

2. **3-layer Convolution Neural Network for Classification**
    1. without the deep learning framework (only python)
    2. using a deep learning framework (e.g. pytorch, tensorflow)

Overall steps

1. Prepare the training and test datasets (MNIST)
2. (For NN) Build a 3-layer Neural Network
   1. Implement a sub-modules (Linear layer, ReLU)
   2. Implement functions (SoftMax, CE loss)
   3. Build a 3-layer NN
3. (For NN) Implement training pipeline & train NN
4. (For NN) Test the 3-layer NN, draw figures
   & Compare results from 1-(1) only python and 1-(2) using frameworks
5. (For CNN) Build a 3-layer CNN & train CNN
   1. Design a Conv layer, Pooling layer & their backpropagation
   2. Build a 3-layer CNN by replacing Linear layer to Conv layer
   3. Train 3-layer CNN
6. (For CNN) Test the 3-layer CNN, draw figures, then compare to NN
   & Compare results from 2-(1) only python and 2-(2) using frameworks

Refer: https://github.com/suqi/deeplearning_andrewng/blob/master/Course1-DL-basic/Week%204/Deep%20Neural%20Network%20Application:%20Image%20Classification/Deep%20Neural%20Network%20-%20Application%20v3.ipynb

1. Prepare training/test dataset

1. **Download MNIST datasets & data loader (see uploaded file)**
    1. Download link: http://yann.lecun.com/exdb/mnist/
        1. train-images-idx3-ubyte.gz: training set images (9912422 bytes)
           train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
           t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
           t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

2. **Prepare the datasets for training**
    1. Ex) normalization

2. Design 3-layer Neural Network(NN)

1. **Design a sub-modules & their backpropagation**
    1. Linear layer
    2. ReLU

2. **Design functions & their derivatives**
    1. SoftMax
    2. Cross-entropy loss

3. **Design 3-layer Neural Network(NN)**
    1. Sequence: Input – Linear-ReLU – Linear-ReLU – Linear-SoftMax
       (The input and output size of NN: input 28x28, output 10)

3. Implement training pipeline, train NN

1. Implement stochastic gradient descent (SGD)

2. Training pipeline
    1. Initialize the model parameters
    2. Implement and do forward propagation
    3. Implement and compute the cross-entropy loss
    4. Implement and do backward propagation
    5. Implement and update model parameter using SGD

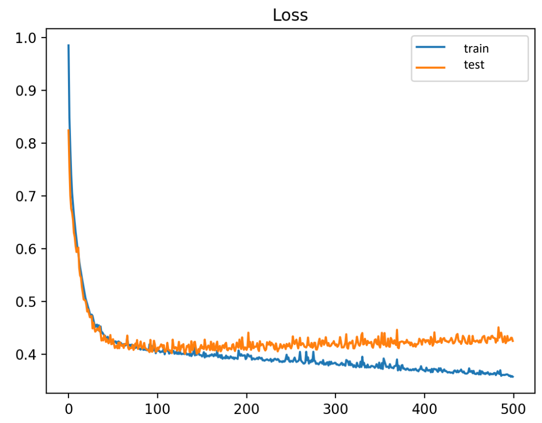
3. Train a 3-layer NN

4. Test the CNN, draw figures

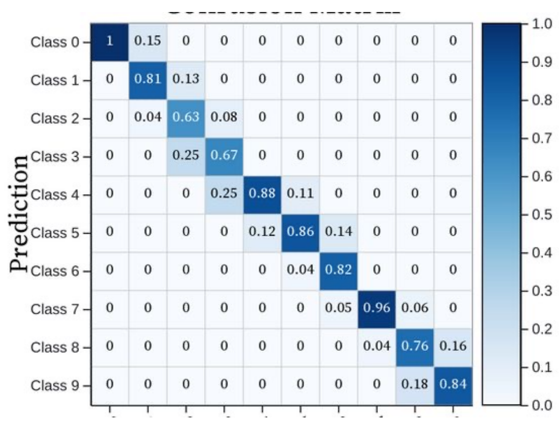
Draw all the output of NN, then write them in a report

1. Show training Loss graph **(Train & test set)**
2. Show 10x10 confusion matrix (the probability matrix of classification)
3. Show top 3 scored images with probability (for each class)

1. Loss graph



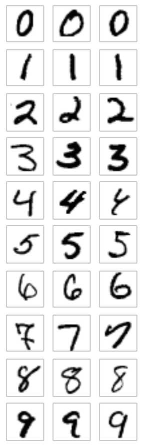
2. Confusion matrix



Class 0	1	0.15	0	0	0	0	0	0	0	0
Class 1	0	0.81	0.13	0	0	0	0	0	0	0
Class 2	0	0.04	0.63	0.08	0	0	0	0	0	0
Class 3	0	0	0.25	0.67	0	0	0	0	0	0
Class 4	0	0	0	0.25	0.88	0.11	0	0	0	0
Class 5	0	0	0	0	0.12	0.86	0.14	0	0	0
Class 6	0	0	0	0	0	0.04	0.82	0	0	0
Class 7	0	0	0	0	0	0	0.05	0.96	0.06	0
Class 8	0	0	0	0	0	0	0	0.04	0.76	0.16
Class 9	0	0	0	0	0	0	0	0	0.18	0.84

3. Top-3 images with probability

0: 1: 2: 3: 4: 5: 6: 7: 8: 9:



100%, 99.8%, 99.7%

⋮

100%, 99.9%, 99.9%

5. Build a CNN by replacing Linear layer to Conv layer

1. **Design a sub-modules & their backpropagation**
    1. Conv layer
    2. Max Pooling

2. **Design CNN**
    1. Sequence: Input – Conv–ReLU–MaxPooling – Conv–ReLU–MaxPooling – Linear–SoftMax
       (The input and output size of CNN: input 28x28, output 10)

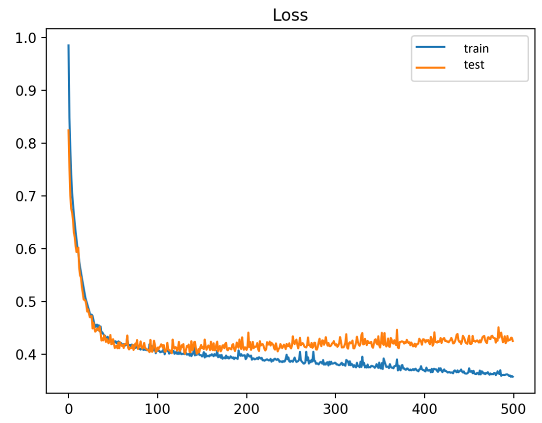
3. **Train CNN**

6. Test the CNN, draw figures

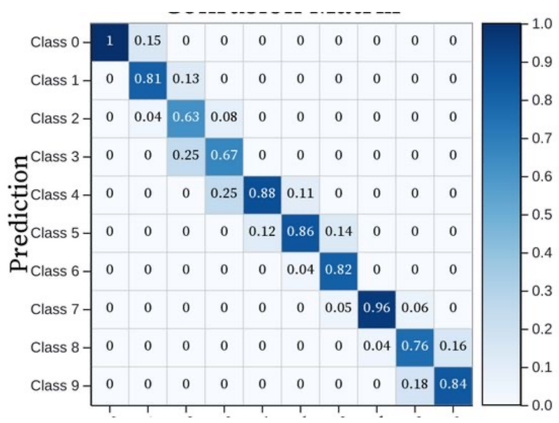
Draw all the same figures and compare with the output of NN

1. Show training Loss graph **(Train & test set)**
2. Show 10x10 confusion matrix (the probability matrix of classification)
3. Show top 3 scored images with probability (for each class)

1. Loss graph

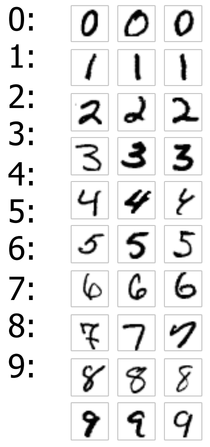


2. Confusion matrix



Class 0	1	0.15	0	0	0	0	0	0	0	0
Class 1	0	0.81	0.13	0	0	0	0	0	0	0
Class 2	0	0.04	0.63	0.08	0	0	0	0	0	0
Class 3	0	0	0.25	0.67	0	0	0	0	0	0
Class 4	0	0	0	0.25	0.88	0.11	0	0	0	0
Class 5	0	0	0	0	0.12	0.86	0.14	0	0	0
Class 6	0	0	0	0	0	0.04	0.82	0	0	0
Class 7	0	0	0	0	0	0	0.05	0.96	0.06	0
Class 8	0	0	0	0	0	0	0	0.04	0.76	0.16
Class 9	0	0	0	0	0	0	0	0	0.18	0.84

3. Top-3 images with probability



100%, 99.8%, 99.7%

⋮

100%, 99.9%, 99.9%

Submission

Due : Oct 16, 14:30PM
To : lms.dgist.ac.kr

1. **Submit zip file including (1) Source code, (2) PDF file(report)**
   1. File name: PA1_studentID_name.zip (PA1_202412345_이경민.zip)
   2. Training & Test datasets should be excluded (only code and report)
2. **Report should include the results and results comparisons:**
   1. Results from all networks (1) NN with python, (2) NN with DL framework, (3) CNN with python, and (4) CNN with DL framework should be attached.
   2. Results mean:
      (a) Training Loss graph
      (b) 10x10 Confusion Matrix
      (c) Top 3 score images (all classes)

   * Report does not need to include your understanding (Results are important)

3. **Final credit: 30% of total grade**

– TA: Gihyun Baek (gh.baek@dgist.ac.kr)

Notice

Due : Oct 16, 14:30PM
To : lms.dgist.ac.kr

1. **Library**
   1. For the PA without the deep learning framework, you cannot use any libraies from Tensorflow, Pytorch, etc, but you can use Numpy or other libraries.
2. **Delayed submission**
   1. 25% score will be degraded every 1-day delay & after 3 days delayed, you will get 10% of the total score & after 1 week delayed, 0%.
      * 100% → 75% (1day) → 50% (2days) → 25% (3days) → 10% (3~7days) → 0% (〉 7days)
3. **Plagiarism**
   1. No grade for copied codes (from friends and the internet)
   2. You can refer to sources from the internet, but do not copy and paste.
   3. You can use LLMs (e.g. ChatGPT, Claude, Gemini etc).
4. **Partial credit**
   1. Even though you are not successfully designing the network and obtained reasonable results, please send your code.
   2. There will be partial credit for each module implementation.