MNIST 참고자료

19.04.01

NLP LAB,
Department of Computer Engineering,
Kyung Hee University.

Index

- What is MNIST ?
- How to download MNIST data
- Data description

What is MNIST ? (1/1)

- MNIST
  - Modified National Institute of Standards and Technology database
  - The digits have been size-normalized and centered in a fixed-size image (28 x 28)
  - taken from NIST's training dataset

  - An extended dataset similar to MNIST called EMNIST has been published in 2017

Download MNIST data (1/2)

- Website
  http://yann.lecun.com/exdb/mnist/

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

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)

Download MNIST data (2/2)

- Code (using by Python 2.x or 3.x)

```
> pip install tensorflow
```

```
>>> from tensorflow.examples.tutorials.mnist import input_data
>>> input_data.read_data_sets("YOUR_FOLDER/", one_hot=True)
```

```
1  # -*- coding: utf-8 -*-
2
3  # MNIST 데이터를 다운로드 한다.
4  from tensorflow.examples.tutorials.mnist import input_data
5  mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
6
```

Data description (1/)

- These files are not in any standard image format
- The data is stored in a very simple file format designed for storing vectors and multidimensional matrices.
- All the integers in the files are stored in the MSB first (high endian) format used by most non-Intel processors.
- Users of Intel processors and other low-endian machines must flip the bytes of the header.

Data description (1/)

- The training set contains 60000 examples, and the test set 10000 examples.

- The first 5000 examples of the test set are taken from the original NIST training set. The last 5000 are taken from the original NIST test set. The first 5000 are cleaner and easier than the last 5000.

- Result 60000 examples
  = 55000 train data + 5000 validation data

Data description (1/)

File	Purpose
`train-images-idx3-ubyte.gz`	training set images - 55000 training images, 5000 validation images
`train-labels-idx1-ubyte.gz`	training set labels matching the images
`t10k-images-idx3-ubyte.gz`	test set images - 10000 images
`t10k-labels-idx1-ubyte.gz`	test set labels matching the images

DataSet Object

The underlying code will download, unpack, and reshape images and labels for the following datasets:

Dataset	Purpose
`data_sets.train`	55000 images and labels, for primary training.
`data_sets.validation`	5000 images and labels, for iterative validation of training accuracy.
`data_sets.test`	10000 images and labels, for final testing of trained accuracy.

Data description (1/)

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

Pixels are organized row-wise. Pixel values are 0 to 255. 0 means background (white), 255 means foreground (black).

Data description (1/)

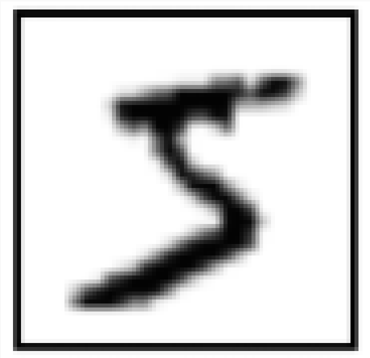
- Image example

train-images.idx3-ubyte

```
1  0000 0803 0000 ea60 0000 001c 0000 001c
2  0000 0000 0000 0000 0000 0000 0000 0000
3  0000 0000 0000 0000 0000 0000 0000 0000
4  0000 0000 0000 0000 0000 0000 0000 0000
5  0000 0000 0000 0000 0000 0000 0000 0000
6  0000 0000 0000 0000 0000 0000 0000 0000
7  0000 0000 0000 0000 0000 0000 0000 0000
8  0000 0000 0000 0000 0000 0000 0000 0000
9  0000 0000 0000 0000 0000 0000 0000 0000
10 0000 0000 0000 0000 0000 0000 0000 0000
11 0000 0000 0000 0000 0312 1212 7e88 af1a
12 a6ff f77f 0000 0000 0000 0000 0000 0000
13 1e24 5e9a aafd fdfd fdfd e1ac fdf2 c340
```

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
0000000000	00	00	08	03	00	00	ea	60	00	00	00	1c	00	00	00	1c
0000000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000a0	00	00	00	00	00	00	00	00	03	12	12	12	7e	88	af	1a
00000000b0	a6	ff	f7	7f	00	00	00	00	00	00	00	00	00	00	00	00
00000000c0	1e	24	5e	9a	aa	fd	fd	fd	fd	fd	e1	ac	fd	f2	c3	40
00000000d0	00	00	00	00	00	00	00	00	00	00	00	31	ee	fd	fd	fd
00000000e0	fd	fd	fd	fd	fd	fb	5d	52	52	38	27	00	00	00	00	00
00000000f0	00	00	00	00	00	00	00	12	db	fd	fd	fd	fd	fd	c6	b6
0000000100	f7	f1	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000110	00	00	00	00	50	9c	6b	fd	fd	cd	0b	00	2b	9a	00	00
0000000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000130	00	0e	01	9a	fd	5a	00	00	00	00	00	00	00	00	00	00
0000000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	8b
0000000150	fd	be	02	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000160	00	00	00	00	00	00	00	00	00	00	00	0b	be	fd	46	00
0000000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Data description (1/)



1 번째 : b'\x05' -> 5

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	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	14	1	154	253	90	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	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	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	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	252	253	187	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	249	253	249	64	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	23	66	213	253	253	253	253	198	81	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	18	171	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	55	172	226	253	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	136	253	253	253	212	135	132	16	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

Data description (1/)

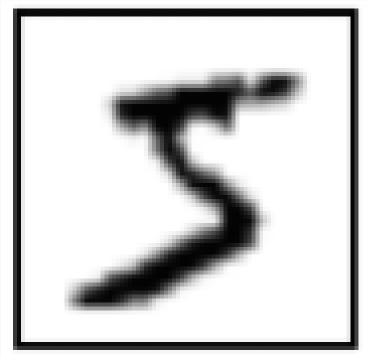
- Label example

train-images.idx3-ubyte × train-labels.idx1-ubyte ×

```
1  0000 0801 0000 ea60 0500 0401 0902 0103
2  0104 0305 0306 0107 0208 0609 0400 0901
3  0102 0403 0207 0308 0609 0005 0600 0706
4  0108 0709 0309 0805 0903 0300 0704 0908
5  0009 0401 0404 0600 0405 0601 0000 0107
6  0106 0300 0201 0107 0900 0206 0708 0309
7  0004 0607 0406 0800 0708 0301 0507 0107
8  0101 0603 0002 0903 0101 0004 0902 0000
9  0200 0207 0108 0604 0106 0304 0509 0103
10 0308 0504 0707 0402 0805 0806 0703 0406
11 0109 0906 0003 0702 0802 0904 0406 0409
12 0700 0902 0905 0105 0901 0203 0203 0509
13 0107 0602 0802 0205 0007 0409 0708 0302
14 0101 0803 0601 0003 0100 0001 0702 0703
```

```
          00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
0000000000 00 00 08 01 00 00 ea 60 05 00 04 01 09 02 01 03
0000000010 01 04 03 05 03 06 01 07 02 08 06 09 04 00 09 01
0000000020 01 02 04 03 02 07 03 08 06 09 00 05 06 00 07 06
0000000030 01 08 07 09 03 09 08 05 09 03 03 00 07 04 09 08
0000000040 00 09 04 01 04 04 06 00 04 05 06 01 00 00 01 07
0000000050 01 06 03 00 02 01 01 07 09 00 02 06 07 08 03 09
0000000060 00 04 06 07 04 06 08 00 07 08 03 01 05 07 01 07
0000000070 01 01 06 03 00 02 09 03 01 01 00 04 09 02 00 00
0000000080 02 00 02 07 01 08 06 04 01 06 03 04 05 09 01 03
0000000090 03 08 05 04 07 07 04 02 08 05 08 06 07 03 04 06
00000000a0 01 09 09 06 00 03 07 02 08 02 09 04 04 06 04 09
00000000b0 07 00 09 02 09 05 01 05 09 01 02 03 02 03 05 09
00000000c0 01 07 06 02 08 02 02 05 00 07 04 09 07 08 03 02
00000000d0 01 01 08 03 06 01 00 03 01 00 00 01 07 02 07 03
```

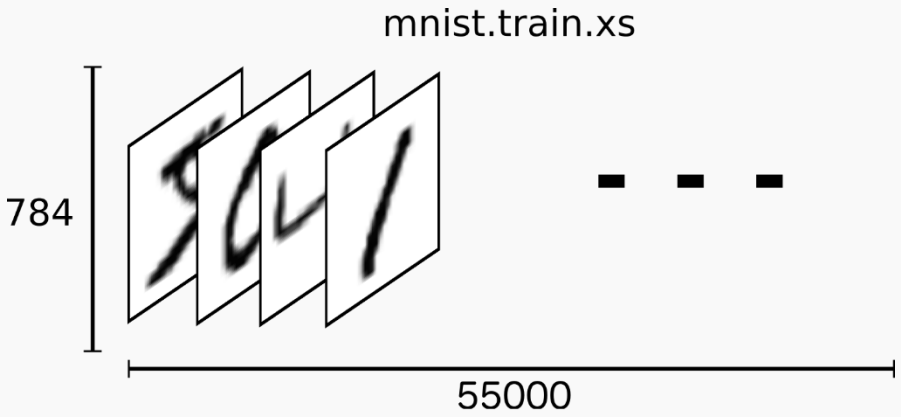
Data description (1/)



```
1 번째 : b'\x05' -> 5
  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   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0
  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0
  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0
  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0 136 253 253 253 212 135 132  16   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
```

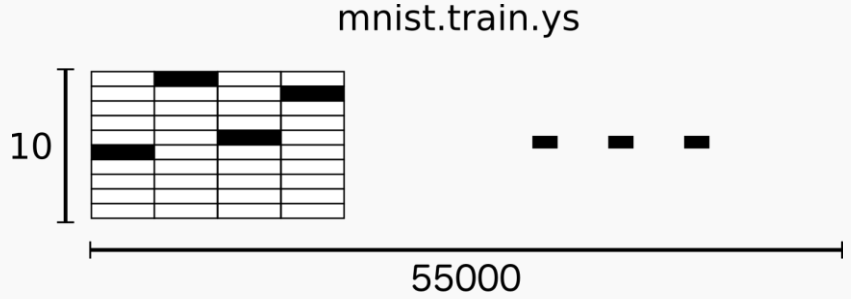
Data description (1/)

- We can flatten this array into a vector of 28x28 = 784 numbers.
- The result is that mnist.train.images is a tensor (an n-dimensional array) with shape (55000L, 784L).



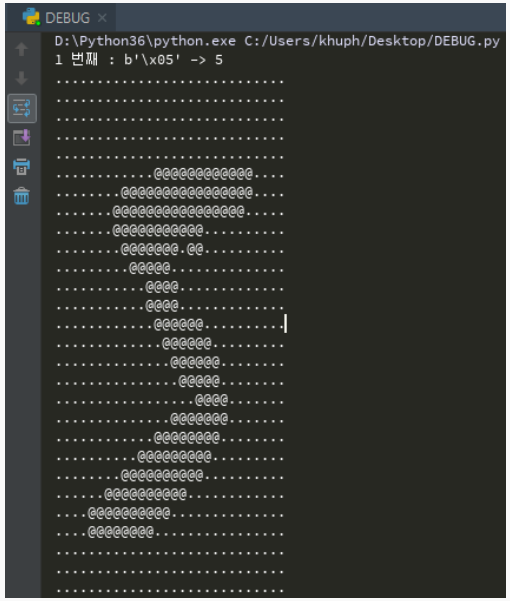
Data description (1/)

- Each image in MNIST has a corresponding label, a number between 0 and 9 representing the digit drawn in the image.
- A one-hot vector is a vector which is 0 in most dimensions, and 1 in a single dimension.
- Consequently, mnist.train.labels is a (55000L, 10L) array of floats.



Data description (1/)

- Sample Code
  https://github.com/DevHyung/Util-module/blob/master/mnist_convert.py




```python
def convert_byte_to_image(imgPath :str, labelPath : str):
    '''
    :param imgPath: img ubyte file path
    :param labelPath:  label ubyte file path
    :return: None
    '''
    imgF = open(imgPath, "rb")
    labelF = open(labelPath, "rb")
    loopIdx : int = 1

    # Skip Byte separately
    imgF.read(16) # Img Ubyte file 16Byte skip
    labelF.read(8) # Label Ubyte file 8Byte skip

    # Read
    labelByte: bytes = b''
    imgByte: bytes = b''
    while True:
        labelByte = labelF.read(1)
        if labelByte == b'':
            break
        print("{} 번째 : {}".format(loopIdx, "{} -> {} ".format(labelByte,ord(labelByte))))
        for i in range(28):
            for j in range(28):
                imgByte = imgF.read(1)
                # If you want 0-255 value, you need to using the ord() function
                # print(str(ord(imgByte)).center(3),end = ' ')
                if imgByte == b'\x00':
                    print(".", end='')
                else:
                    print("@", end='')
            print()
        loopIdx +=1
        _ = input(">>> Next :")
        os.system("cls")

    imgF.close()
    labelF.close()


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('--imgfile', type=str, default="train-images.idx3-ubyte",
                        help='Mnist Train or Test img file path')
    parser.add_argument('--labelfile', type=str, default="train-labels.idx1-ubyte",
                        help='Mnist Train or Test label file path')
    args = parser.parse_args()
    _imgFilePath : str = str(args.imgfile)
    _labelFilePath : str = str(args.labelfile)
    # Function Call
    convert_byte_to_image(imgPath=_imgFilePath,labelPath=_labelFilePath)
```

Q & A