MNIST handwritten digits Description and using

Technical Report · November 2014

DOI: 10.13140/2.1.4601.1681

1 author:

Mazdak Fatahi

Razi University

8 PUBLICATIONS **2** CITATIONS

SEE PROFILE

MNIST handwritten digits

Description and using

Mazdak Fatahi

Oct. 2014

Abstract

Every scientific work needs some measurements. To compare the performance and accuracy of handwriting recognition methods which innovated, the MNIST dataset is a very good dataset consists of 60,000 samples for training and 10,000 test samples. In this report we provide a description on MNIST dataset and present some MATLAB codes for reading the IDX file format, which MNIST samples are stored in this file format.

1- MNIST database

There are some famous databases for examining the accuracy and performance of algorithms proposed by machine learning researchers and professionals. These databases are in various fields. For example if someone needs to examine his new approach in neural networks for recognition of facial expressions, he needs a lot of train data and should has some data for test. Also if he wants to compare his work with others, he must use a database similar to others. For this problem there are some standard databases. One of this databases in handwritten digits recognition is MNIST. MNIST is a standard and large database of handwritten digits.

MNIST dataset has been widely used as a benchmark for testing classification algorithms in handwritten digit recognition systems [1]. MNIST is an abbreviation for Mixed National Institute of Standards and Technology database. This database is created by "re-mixing" the original samples of NIST's database. The database has two parts: *training samples* that was taken from American Census Bureau employees and the *test samples* that was taken from American high school students.

The original NIST's database is too hard, therefore the MNIST database was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits. The samples that was taken from American Census Bureau employees (training database), was very cleaner than the *samples* that was taken from American high school students (test database). It can makes dataset dependency for our learning algorithm (because of more readability of training set than test set). Therefore ''re-mixing'' NIST's datasets was necessary [2].

By combining of 30,000 samples from first dataset and 30,000 samples from second dataset, the Mixed NIST training set was created. Also 60'000 test samples was collected to constitute the test dataset, but only 10,000 of patterns is now available on MNIST webpage. 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) [2].

The MNIST is widely used for training and testing in the field of machine learning. In some papers the training set was extended by adding some distortions (random combinations of scaling, shifts, adding some noise etc.). In [2] there are the results of some methods that have been tested with MNIST dataset.

The MNIST database is available on MNIST webpage[1] and contains of 60,000 examples for training set and 10,000 examples for test set. This database includes 4 files:

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)

[1] http:// yann.lecun.com/exdb/mnist/ - The home of the database

2- MNIST's file format

In MNIST database the original black and white images from NIST were size normalized to fit in a 20x20 pixel box. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm and the images were centered in a 28x28 [2].

The data of all images in each database are joined in a single file and are stored in a file format that called IDX. IDX is a file format for sorting vectors and multidimensional matrices of various numerical types. The basic format is:

magic number
size in dimension 0
size in dimension 1
size in dimension 2
.....
size in dimension N
data

The magic number is an integer that stored in the MSB first format, then for users of Intel processors and other low-endian machines must flip the bytes of the header.

The magic number format is:

2 Bytes	1 Byte	1 Byte
0 0	Data type	Dimensions#

Data type	Dimensions#
0x08: unsigned byte 0x09: signed byte 0x0B: short (2 bytes) 0x0C: int (4 bytes) 0x0D: float (4 bytes) 0x0E: double (8 bytes)	1 for vectors 2 for matrices 3 for 3-D matrices . . .

For example about first dataset file:

File name:
train-images-idx3-ubyte.gz: training set images (9912422 bytes)

[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 white and 255 means black.)

All the numbers are sequentially stored in a binary file. The offset shows the beginning of each part of data. For example the magic number is started at 0000, that mean the beginning of the file. The magic number was shown in Tabel.1.

Tabel.1. Magic Number bytes description

2 Bytes	Data type(1 Byte)	Dimensions#(1 Byte)
00 00	08	03

That means the matrix has 3 dimensions, and the data type is (08) unsigned byte.

The next rows, from the IDX format, are the number of elements in each dimensions. The number of each dimensions are shown in 4 bytes (32 bit integer):

0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns

That means the first dimension has 60,000 elements and each of them has 28*28 elements (which are the pixels of each image).

3- Using MATLAB to extract the images from MNIST Files

From the IDX file format and by using of the previous guidelines, it is possible to extract the data of *Images* and *labels* from MNIST's files.

3-1- Reading images

After downloading and decompressing, we have 4 row binary files. For train-images-idx3-ubyte dataset, from the IDX format, the data of pixels are started at byte 16. For each image from 60,000 images, there are 28*28 pixels which are organized row-wise. Then from offset 16 to 800 (16+28*28), there are the pixels of first image in dataset.

These pixels can be read as follow (In MATLAB) [3]:

```
trainImg = fopen('train-images-idx3-ubyte','r','b'); % first we have to open the binary file

MagicNumber=fread(trainImg,1,'int32')

MagicNumber =

        2051

nImages= fread(trainImg,1,'int32')% Read the number of images

nImages =

        60000

nRows= fread(trainImg,1,'int32')% Read the number of rows in each image
```

```
nRows =

        28


nCols= fread(trainImg,1,'int32')% Read the number of columns in each image

nCols =

        28
```

Note: these steps are not required, because we know the number of elements in each dataset, and we know the properties of each element.

Pixels of images are stored at 16th byte to end, and each 784(28*28) bytes are pixels of one image which is stored row-wise. Then we need to reshape the vector of 784 pixels into a 28*28 matrix.

We can read 16 byte as header information (as previously done) or we can seek to 16th byte of file (by fseek command in MATLAB) and then start reading the data of images.

```
fseek(trainImg,16,'bof');

img= fread(trainImg,28*28,'uchar');% each image has 28*28 pixels in unsigned byte format
```

The recent line can read 784 bytes from dataset as a vector (img is a vector of 784*1). The digits are stored in row-wise then the pixels of first row are stored in 1 to 28, for second row in 29 to 56 and ….

Then we can reconstruct the digits by reshaping the recent vector into a 28*28 matrix:

```
img2=zeros(28,28);

for i=1:28

        img2(i,:)=img((i-1)*28+1:i*28);

end

imshow(img2);
```

The last command shows the first digit (Fig.1.) in MNIST train data set:

Fig.1. The first MNIST train digit

Because original images from NIST were size normalized by MNIST to fit in a 20x20 pixel and the resulting images contain grey levels (as a result of the anti-aliasing technique used by the normalization algorithm) and also were centered in a 28x28 image, therefore we can trim the extracted image and normalize it to values ranging from [0; 1]:

```
img1(:,:)=imgtmp(5:24,5:24);%fig.2
```

Fig.2. The first MNIST train digit after trimming

img=img1./255;

Fig.3. The first MNIST train digit after trimming and normalizing into [0,1]

3-2- Reading labels

For the label file we can do same:

```
trainlbl = fopen('train-labels-idx1-ubyte','r','b'); % first we have to open the binary file

MagicNumber=fread(trainlbl,1,'int32')

MagicNumber =

        2049

nLabels= fread(trainlbl,1,'int32')% Read the number of labels

nLabels=

        60000
```

Directly from [2] we know the first label is stored in 8th byte of file in unsigned byte format:

```
fseek(trainlbl,8,'bof');

fread(trainlbl,1,'uchar')

ans =

        5
```

4- Extracted images

Finally the result of reading the first 100 images and label shown in the Fig.3.

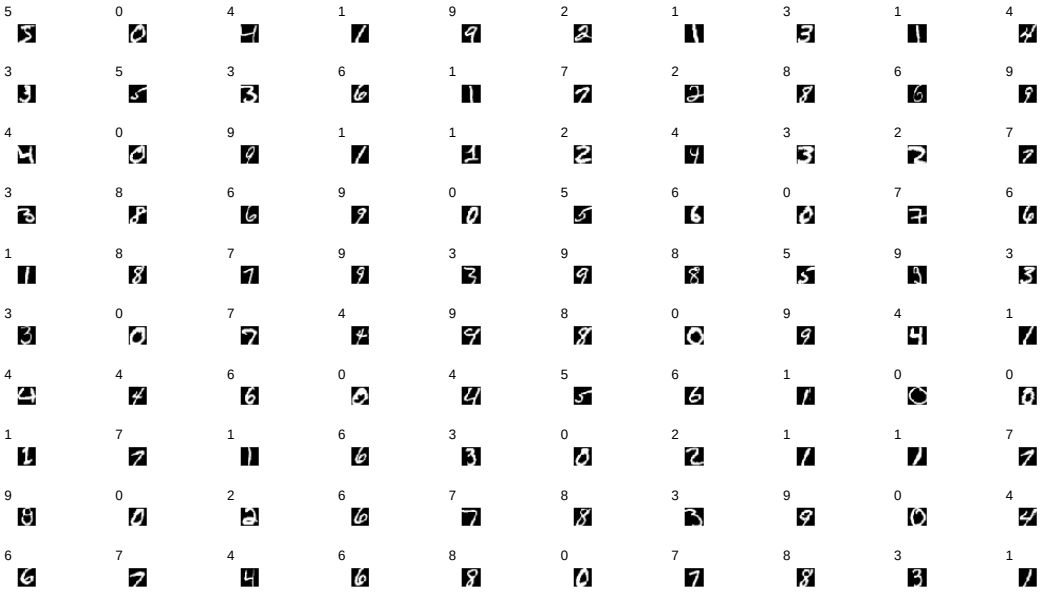

Fig.4. The first 100 samples in MNIST handwritten digits dataset

References

[1] M. Wu and Z. Zhang, "Handwritten Digit Classification using the MNIST Data Set," 2010.

[2] Y. LeCun, C. Cortes and C. J. Burges, "Yann LeCun's Home Page," [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed 29 Oct. 2014].

[3] R. Salakhutdinov and G. Hinton, "Training a deep autoencoder or a classifier on MNIST digits," [Online]. Available: http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html. [Accessed 29 Oct. 2014].