KxCon2016 Programming Challenge[1]

Nick Psaris

19 May 2016

[1] Inspired by Andrew Ng's Machine Learning Coursera Class `http://class.coursera.org/ml-005/lecture`

This challenge was chosen because it can be quickly implemented inefficiently and then considerably optimized. This is not a toy problem – the resulting function is used to load datasets for machine learning. Finally, to make the problem more interesting, an existing `q` operator has been extended[2] in `kdb+ 3.4t` that can make your solution even shorter.

[2] `"reshape extended to >2 dimensions..."`

Machine Learning

Background

A popular application of machine learning is character recognition. If we assume a handwritten digit can be digitized into a vector of pixels, logistic regression (among many other techniques) can be used to assign a weight to each pixel. These learned weights can then be combined with a new image to make a prediction of which digit it represents.

The MNIST[3] database holds a collection of handwritten digits that have been normalized for use in testing machine learning and pattern recognition techniques.

[3] `http://yann.lecun.com/exdb/mnist`

Challenge

To process these images of handwritten digits, we must load the data from files stored in the custom MNIST binary format. Your challenge is to write a function to read this data and return the resulting n-dimensional array. Lucky for you, this format has been well documented on the MNIST site.

The site specifies the exact dimension and numerical type of each dataset. This would allow you to write a custom loader for each file. The file format, however, is self-describing. You are required, therefore, to write a general loader that works with datasets of all dimensions and types. While you are waiting for the dataset to download, you can begin testing your implementation against the unit tests below.

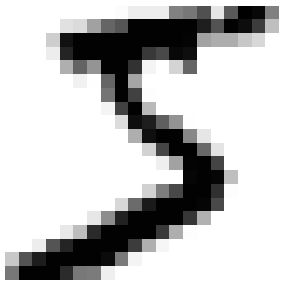

Figure 1: The first image in the MNIST training file representing the number 5.

Rules

Interface

Your function will be applied to the MNIST training dataset.[4] To make the function more flexible, its should accept a byte-vector instead of a file name. The function can then be applied to unit tests to confirm proper behavior. To be accepted, your function named `ldidx` should produce the following results.[5] NOTE: ignore any extra trailing bytes.

[4] `http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz`

[5] signed and unsigned bytes should both be returned as type "x"

```
q)ldidx 0x0000080100000000
`byte$()
q)ldidx 0x000008010000000100
,0x00
q)0N!ldidx 0x00000802000000020000000020001020304;
(0x0001;0x0203)
q)0N!ldidx 0x0000080300000002000000020000000200010203040506070 8;
((0x0001;0x0203);(0x0405;0x0607))
q)ldidx 0x00000b010000000200010002
1 2h
q)ldidx 0x00000c0100000002000000010000000 2
1 2i
q)ldidx 0x00000d01000000023f80000040000000
1 2e
q)ldidx 0x00000e01000000023ff0000000000000400000000000000 0
1 2f
q)md5 raze over string X:ldidx b:read1 `$"train-images-idx3-ubyte"
0x6a5cde79f049959f93df34292c599c1b
```

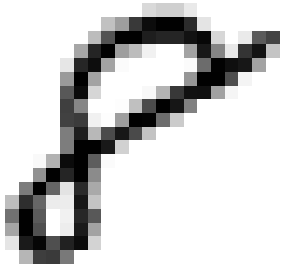

Figure 2: The last image in the MNIST training file representing the number 8.

Submission

Email[6] your function as soon as it produces valid results, and again when you've optimized the code. No external user-defined functions or data structures can be used. Only the first and last submission by an individual will be accepted for the competition. All submissions must be made prior to 00:00 EST on 22 May 2016. The 32 bit free version of q[7] available on 20 May 2016 will be used to test each submission.

[6] nick@vector-sigma.com

[7] `https://kx.com/software-download.php`

Scoring

One point will be awarded for each of the following categories.

1. Fastest valid submission measured in milliseconds elapsed[8]
2. Smallest valid submission measured in allocated bytes[9]
3. Shortest valid submission measured in bytes[10]

[8] q)\t:10 ldidx b

[9] q)\ts ldidx b

[10] q)**count first get** ldidx

In case of a tie, the submitter who provided the first valid submission (irrespective of performance) will win.