18-794 Pattern Recognition Hackathon

Nov. 25, 2016

Dipan K. Pal and T. Hoang Ngan Le (Nancy)
Carnegie Mellon University
dipanp@andrew.cmu.edu, thihoanl@andrew.cmu.edu

1. Overview

The aim of the 18794 Pattern Recognition Hackathon is to introduce students to a real-world pattern recognition problem. This Hackathon assumes a medium level of familiarity with writing MATLAB scripts and is worth 10% of your grade for the course. In order to do well in this Hackathon, you will be required to:

- Implement an end-to-end pattern recognition engine in MATLAB
- Make intelligent decisions on selecting pre-processing methods, features, and classifiers
- Learn new ways to evaluate your classifier
- Think out of the box!

2. Problem Setup

In this Hackathon, your goal is to implement a digit classification algorithm using the MNIST dataset as shown in Figure 1. The MNIST project website is http://yann.lecun.com/exdb/mnist/.

2.1. Data

The MNIST website has provided 60,000 training images, and 10,000 testing images, with their corresponding class labels. This is a 10-class classification problem. Although the training and testing data are already provided on the website,

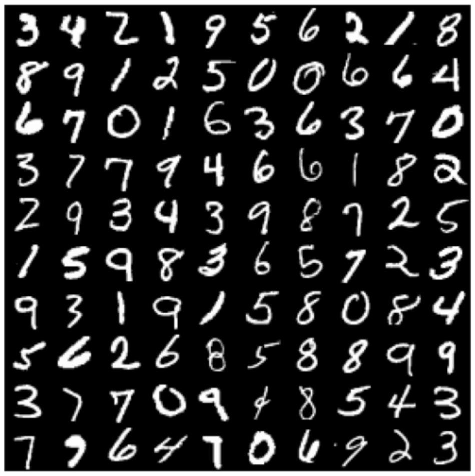

Figure 1: Examples from the MNIST digits dataset.

we request that you use the training / testing data we provide in .MAT format. The data is in `double` format, whose values are between 0 and 1. You can download the data from the Blackboard, under Assignment/Hackathon/.

2.1.1 Training Data

In addition to using all 60,000 training images, we also want to see how well your algorithm can perform when given limited amount of training data. Therefore, we have provided 3 training data files, each containing 60,000, 12,000, and 3,000 training images respectively as follows:

- `train_60k_mnist.mat`
- `train_12k_mnist.mat`
- `train_3k_mnist.mat`

Inside each of the .MAT file, there are two variables: `imgs`, which is a 3D array containing the training images, each of size 20×20, and `labels`, which is a vector containing the class labels corresponding to the 3D array in the `imgs`. Notice that, in each training corpus, there may not be equal number of training images for each of the 10 classes.

2.1.2 Testing Data

Similar to the training data, the 10,000 testing images and their labels are stored in:

- `test_10k_mnist.mat`

Your algorithm should output classification results for each of the 10,000 testing images, and the final error rate is computed as:

$$\begin{aligned} \texttt{err}\% &= \frac{N \times 100}{10,000}\% \\ \texttt{err} &= \frac{N \times 100}{10,000} \end{aligned} \tag{1}$$

where N is the number of testing images wrongly classified.

3. Duration

This Hackathon begins at **Monday Nov. 25, 2016 at 8:00am**, and ends at **Friday Dec. 9, 2016 at 8:00am**. We have given you a two-week duration to account for your flexibility. That being said, do not over-stress yourselves.

4. Submission

Submission will include your entire training and testing code, as well as a write-up describing your algorithm. We should be able to reproduce your results on our machine. In special cases, if you use any fancy libraries, we might schedule a separate meet up to evaluate and run your code. Since you can locally evaluate the performance of your classification algorithm, we suggest you tune your algorithms **on a cross-validation dataset (part of the training data ONLY)** to achieve lowest error rate as possible, and then submit it towards the end of the Hackathon. Submission will be either through Blackboard, or email. Details will be announced as the Hackathon progresses.

4.1. Code

We should be able to re-run your training and testing code, and produce the same results you have reported in the write-up. For submission in the Hackathon, you will submit one zip file named `18794_hackathon_andrewid.zip`, where `andrewid` is your CMU Andrew ID. The zip file should contain the following:

- **run_training.m**: This script file calls 3 functions with the following form:

  ```
  function [model_60k] = mnist_train_60k(imgs, labels);
  function [model_12k] = mnist_train_12k(imgs, labels);
  ```

```
function [model_3k] = mnist_train_3k(imgs, labels);
```

`model_60k, model_12k, model_3k` each is a struct which should contain all the necessary parameters trained using corresponding training corpus for your digit classifier which will then be passed as the third, fourth, and fifth argument (in testing code) to classify MNIST digits.

IMPORTANT: Note that you are allowed to have a different model for each training dataset size. You are required to perform any hyper-parameter tuning ONLY on a validation set you set aside from each of the training sets provided. You are NOT allowed to tune any hyper-parameters using the test set. Your hyper-parameter choices would be verified. You should clearly explain in your write up as to why you chose the hyper-parameter values you did. This should be the one which gives the best cross-validation accuracy for each training set size.

- **run_testing.m**: This script file calls a function with the following form:

  ```
  function [err_60k, err_12k, err_3k] = mnist_test_10k(imgs, labels, model_60k, model_12k,
  model_3k);
  ```

  The `errs` follows what is defined in (1).

- Supplementary MATLAB function files (*.m), MATLAB data files (*.mat) containing parameters, etc. and library DLLs, executables, etc. as required by the above code.

- **algorithm.txt** : A text file with a brief description of the algorithm in the submission. If any external libraries are used in this submission, the purpose of these libraries and the need for using these libraries should be clearly stated. All fixed parameters used in this submission (such as thresholds, weights, etc.) should be explained, including how the submitted values were arrived at.

4.2. Write-up

Along with the code submission `18794_hackathon_andrewid.zip`, a short report is also required. The report should be named `18794_hackathon_andrewid.pdf`. The write-up should be 1-2 pages with a detailed explanation of the algorithms in the submission, as opposed to the brief explanation in the **algorithm.txt**. It should also contain the error rates (`err_60k, err_12k, err_3k`) and 10 failure cases for each of the 3 training paradigms.

5. Scoring Scheme

You will report (in your write up) your error rates on the testing dataset (10,000 images) for each of the models trained on the training sets (3K, 12K and 60K). All of the three errors will be used grading.

Your submission of each error rate for each of the training sets of this Hackathon will be graded out of 100. Each of the three models would be graded as follows: The top performer will be given 100 points. The worst performer will be awarded 80 points. This defines a scale from 80 to 100 within which all of the performers would be placed using a linear map. As mentioned, this is done for each of the training set size (dataset). Your final score will be the average of the three scores you get for the three dataset sizes.

Note that we are now not grading you against a baseline to allow for more flexibility. That said, please use the baseline as an indication of how your model is performing on the full 60 K dataset.

Non-submission, plagiarism or disqualification from the Hackathon will receive 0 point.

6. Hackathon Rules

1. All decisions made by the course instructor or TAs are final.
2. There are no late days for this Hackathon.
3. Participants may discuss different strategies among themselves, but each submission should be the direct result of the **individual** student's effort. The plagiarism rules outlined in the course syllabus are imposed in this Hackathon as well.
4. Any submitted code which does not directly fall within the intent of this Hackathon is also considered grounds for immediate disqualification.
5. Any submission not in the specified format (outlined in Section 4) will be rejected.

6.1. Policy regarding Third Party Libraries

You are allowed to use any third party library you can find as long as you can provide a MATLAB interface and under the condition that the TAs can reproduce your results using your code. In special cases where you use specialized hardware/software you might be required to schedule (post-submission of your code online) a meeting with a TA and have your code verified and run.

7. Hints and Strategy

- Your first step should be to take a look at the data, plot it in MATLAB using `imagesc`, try to get an intuitive feel for what might work.
- Try to research some papers on MNIST digit classification. You might hit gold if you look long enough.
- Discussing different techniques with other students might help you, but be wary since this is, after all, a weak form of a competition.
- Note that your algorithm should run for all three
- Enjoy the Hackathon! Do let the TAs know if you think there is any way this Hackathon could be improved.

More hints, clues and suggestions will be released by the TAs in the form of Blackboard announcements.