Project: Test Accuracy of Using Python and Matlab/Octave Implementation
Goal

- Make sure Python and Matlab implementations give same or similar objective values and test accuracy
- Check the variance of different runs
In running two stochastic gradient implementations, easily they give different objective values or test accuracy.

Many things can cause the difference.

For example,
- data pre-processing
- initial solution
- learning rate
- mini-batch selection
- and others
Now we see why checking the reproducibility of a paper is so difficult.

Fortunately, in our code we tried to use the same settings for both implementations.

You should trace the code to confirm this.

An exception is the initial solution. See details later.

Present your findings and thoughts.

For example, you may want to draw figures of iterations versus objective values and/or iterations versus test accuracy.
Data Sets I

- MNIST is a simple set (test accuracy very high), so let’s consider three more difficult sets 
  SCHN, CIFAR10, smallNORB
- Their .m files are available at LIBSVM data sets
- Test sets are available to check accuracy
Network Architecture

- In the past projects we consider -net CNN_4layers
- Now let’s consider more layers by using CNN_7layers
- This can be directly done for the Python code, but for MATLAB you need to modify the config file accordingly
Pre-processing I

- We explain the pre-processing steps used in our code
- You can trace the code to confirm these steps
- Note that many possible pre-processing steps may be taken. Here we consider some common ones
- Normalization.
  
  For each pixel of every image $Z^{1,i}$, we have
  
  $$Z_{a,b,d}^{1,i} \leftarrow \frac{Z_{a,b,d}^{1,i}}{255},$$

- Zero-centering.
This is commonly applied before training CNN (Krizhevsky et al., 2012; Zeiler and Fergus, 2014). For every pixel in image $Z^{1,i}$, we have

$$Z_{a,b,d}^{1,i} \leftarrow Z_{a,b,d}^{1,i} - \text{mean}(Z_{a,b,d}^{1,:}),$$

where $\text{mean}(Z_{a,b,d}^{1,:})$ is the per-pixel mean value across all the training images.
We follow He et al. (2015) to set the weight values by multiplying random values from the $\mathcal{N}(0, 1)$ distribution and

$$
\sqrt{\frac{2}{n_{\text{in}}^m}}, \quad \text{where } n_{\text{in}}^m = \begin{cases} 
d^m \times h^m \times h^m & \text{if } m \leq L^c, \\
n_m & \text{otherwise.}
\end{cases}
$$

For the bias, let set the initial value to be

$$b = 0$$
To use same initial weights, our strategy is to generate weights from MATLAB and load them from Python.

First, in the MATLAB code, after the initial weights are given, call `save_init(model)` by the file available in this directory.

A file `init.mat` will be generated.
Second, we need to insert a line before the fully-connected layer in net.py. The reason is because of MATLAB’s column-wise versus NUMPY’s row-wise storage. Thus after the last pooling operation, let’s add

```python
pool = tf.transpose(pool, perm=[0, 2, 1, 3])
```

Finally, we modify the generation of initial weights in train.py by replacing

```python
sess.run(init_model(param))
```

with

```python
load_weight(sess, param, 'init.mat')
```
To get the same behavior, it is important to consider the same parameter

- Mini-batch size: we use 128
- We consider the algorithm with momentum
- We use $\alpha = 0.9$

\[ \nu \leftarrow \alpha \nu - \eta \left( \frac{\theta}{C} + \frac{1}{|S|} \nabla \theta \sum_{i:i \in S} \xi(\theta; y^i, Z_{1,i}^i) \right) \]

\[ \theta \leftarrow \theta + \nu \]
We run 500 epochs

Initial learning rates: let’s use 0.003

For the regularization parameter, let’s fix it to be

\[ C = 0.01 \]
Even with exactly the same algorithm, results may still be different due to the sequence of random numbers used.

Note that random numbers are used in:
- generating the initial weights, and
- selecting the mini batch.

To handle this situation, let’s do 5 runs and analyze results.

We hope to achieve that the mean accuracy values of the two implementations (MATLAB and Python) are very similar.
Presentations I

- Details will come soon
If needed we may increase the page limit


Acknowledgments

- Cheng-Hung Liu helped to check many issues related to this project file.
- The setting to have the same initial solution was prepared by Pengrui Quan at UCLA.