Project: An Investigation of Python Profilers

Last updated: May 23, 2021

Introduction I

- Earlier in a project to check the running time of MATLAB and Tensorflow implementations, we could only profile the MATLAB code
- Our Tensorflow code, based on Tensorflow 1.xx, is not procedural
- This seems to make the profiling difficult
- In this project let's consider the package LibMultiLabel at `https://github.com/ASUS-AICS/LibMultiLabel` for multi-label text classification
- It is based on PyTorch and has a procedural setting

Introduction II

- The goal is to do profiling for running this package
- The profiling should be similarly detailed as MATLAB
- The network is simple: one convolutional layer and then one linear layer
- This is also a chance for us to learn how CNN is used for text data
- The package is being actively developed. If changes may affect your projects, we will let you know.

Introduction III

- Anyway, we encourage you to always pull the latest version. For problems related to the package, you can directly file an issue on Github

CNN for Text Data I

- Assume each document has the following word embeddings

$$X = \begin{bmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_N \end{bmatrix} \in R^{d_e \times N},$$

  where d_e is the word-embedding dimension and N is the document length.
- That is, by some ways we have already obtained some information for each word

CNN for Text Data II

- For any filter

  $$\boldsymbol{v} \in R^{d_e \times k},$$

  a convolutional operation is applied to a text region

  $$[\boldsymbol{x}_n, \ldots, \boldsymbol{x}_{n+k-1}] \in R^{d_e \times k}$$

  of k words
- It is like that we treat X as an image and horizontally extract sub-images

CNN for Text Data III

- Thus the following operation is conducted:

$$(h_n)_j = \sigma(\langle W_{1:d_e,1:k,j}, [\boldsymbol{x}_n, \ldots, \boldsymbol{x}_{n+k-1}]\rangle + b_j),$$

where $\boldsymbol{h}_n$ is the nth output vector, $\langle \cdot, \cdot \rangle$ is the component-wise sum of two matrices,

$$W_{1:d_e,1:k,j} \in R^{d_e \times k}$$

is the jth filter, and σ is an activation function.

CNN for Text Data IV

- Here

$$j = 1, \ldots, d_c$$

so d_c is the number of filters.

- The output after the convolutional operation is a matrix

$$H = \begin{bmatrix} \boldsymbol{h}_1 & \ldots & \boldsymbol{h}_{N-k+1} \end{bmatrix} \in R^{d_c \times (N-k+1)}$$

- Assuming the input is not zero-padded

Pooling I

- The maximum from each row of H is collected

$$g_i = \max_j H_{ij}$$

$$\boldsymbol{g} = \begin{bmatrix} g_1 & \dots & g_{d_c} \end{bmatrix}^\mathsf{T} \in R^{d_c}$$

- This naturally allows for variable document length N

Linear I

- The final layer is a linear layer

$$\boldsymbol{z} = A\boldsymbol{g} + \boldsymbol{c} \in R^l$$

where $A \in R^{l \times d_c}$ is the weights, $\boldsymbol{c}$ is the bias and l is the number of classes

Multi-label Prediction I

- Each instance belongs to multiple classes
- Thus we cannot take the maximum of z as the prediction
- For this project, we do not worry about how predictions are done

Cost Analysis I

Convolutional layer

$$k \times d_e \times d_c \times N$$

Pooling layer

$$d_c \times N$$

Linear layer

$$l \times d_c$$

Project Contents I

- See `LibMultiLabel` README for installation instructions
- Let's run 5 epochs on the rcv1 data and do some analysis
- We will use the `kim_cnn` architecture (Kim, 2014)
- Check the example in the "Quick Start via Example" section of README
- In `kim_cnn.yml` you will see a line

```
filter_sizes: [2 4 8]
```

Project Contents II

This means that different filter sizes are considered. Let's change to use the size of 2 only for easier analysis

- A key thing is to check the running time of major operations and see if things agree with the complexity analysis
- In particular, we check the forward process which is implemented by us. In contrast, the backward process is done by PyTorch
- The usage is

Project Contents III

```
python3 main.py --cpu --config \
example_config/rcv1/kim_cnn.yml
```

- For the current setting, the program predicts a test set in the end. Since we are interested only in training, you can remove the test file and the test procedure will not be conducted
- For the training procedure, the code internally splits the training set to 80% for training and 20% for validation
- The validation procedure is used in, for example, deciding when the training procedure should stop

Project Contents IV

- Here we do not need that but there is no option yet to disable the validation procedure.
- This is fine because in your comparison between convolutional and linear layers, they are now both run on 80% of data
- To specify parameters, such as the number of epochs, you need to modify the configuration file `kim_cnn.yml`
- These parameters are specified by the following arguments in the configuration file
    - k: `filter_sizes`

Project Contents V

- d_c: `num_filter_per_size`
- d_e: it depends on `embed_file` (e.g. `glove.6B.300d` is 300)
  Note that for the example we use a pre-trained word embeddings `glove.6B.300d`
- l: 103 for rcv1
- N: average 123.9 for rcv1

Profiler I

- The package `pprofile` provides line by line profiling
- Install it by

  ```
  pip3 install pprofile
  ```

- Basic usage is to replace `python3` with `pprofile`

  ```
  pprofile main.py arg1 arg2 ...
  ```

- No code modification is needed
- Other Python profilers are available, but we found this one useful

Issue of Multiple Cores I

- Let's try both single and multiple cores
- For PyTorch, we do

  ```
  torch.set_num_threads(1)
  ```
- An issue of the above setting is that PyTorch runs 2 threads and uses 50% CPU on each
- We can force a process to use one core by

  ```
  taskset -c 0 [command]
  ```

Optimized BLAS I

- Can we confirm that optimized BLAS is used in PyTorch?

Some Notes on Using 217 Workstations I

- This page provides some help for students who use department workstation.
- Your home directory is unlikely to have enough storage for the embedding file and model data
- You may symlink to /tmp2/$USER

```
mkdir -p /tmp2/$USER/runs
mkdir -p /tmp2/$USER/.vector_cache
ln -s /tmp2/$USER/runs runs
ln -s /tmp2/$USER/.vector_cache .vector_cache
```

- Be sure to read the rules of using /tmp2

References I

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014. doi: 10.3115/v1/D14-1181.