Text Sentiment Analysis with rNN on the IMDB Dataset

PyTorch and TensorFlow Comparative Evaluation

Stefanos Laskaridis
sl829@cam.ac.uk
Recurrent Neural Networks

A. Forward Pass

Let $x_t$ be the input vector at time step $t$, $y_t$ be the output vector, and $c_t$ be the cell value. Then we get the output vector:

$$y_t = o_t h_t$$

where $h_t$ is the hidden state at time step $t$.

The hidden state follows the hidden-to-hidden recurrence:

$$h_t = f_t h_{t-1} + i_t c_t + o_t h_t$$

where $f_t$, $i_t$, and $o_t$ are the forget, input, and output gates, respectively.

The cell value follows:

$$c_t = f_t c_{t-1} + i_t x_t$$

where $c_t$ is the cell state.

Bias weights, peephole weights, and recurrent weights are:

$$f_t = \sigma(W_f x_t + W_{sf} h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + W_{si} h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + W_{so} h_{t-1} + b_o)$$

$$c_t = \tanh(W_c x_t + W_{sc} h_{t-1} + b_c)$$

where $\sigma$ is the hyperbolic tangent function.

The cell state is multiplied with the output gate:

$$h_t = \sigma(W_h x_t + W_{sh} h_{t-1} + b_h)$$

The loss function is formally equivalent to the backpropagation through time (BPTT) algorithm.
Sentiment Analysis

- Make out the general sentiment of a sentence.
- Based word vectors, ngrams, word embeddings, etc.
- Binary or Multiclass classification in different sentiments.

The IMDB dataset

```python
from keras.datasets import imdb

(X_train, y_train), (X_test, y_test) = imdb.load_data()

X = np.concatenate((X_train, X_test), axis=0)
y = np.concatenate((y_train, y_test), axis=0)

print('X shape:', X.shape)
print('y shape:', y.shape)

# Number of classes for classification
print('Dictionary size:', len(np.unique(y)))

# Number of unique words in dataset (dictionary)
print(len(np.unique(np.hstack(X))))

# Avg review length
lengths = [len(r) for r in X]
avg = np.mean(lengths)
stdev = np.std(lengths)
print("length mean:", avg)
print("length stdev", stdev)
```

<table>
<thead>
<tr>
<th>X shape: (50000,)</th>
<th>y shape: (50000,)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary size: 288585</td>
<td></td>
</tr>
<tr>
<td>length mean: 234.75892</td>
<td></td>
</tr>
<tr>
<td>length stdev: 172.911494587</td>
<td></td>
</tr>
</tbody>
</table>
PyTorch vs. TensorFlow

**PyTorch**

- Python implementation of Torch (Lua)
- Imperative programming model
- Great integration with Python
- Beta version
- CPU, GPU
- Facebook, CMU, Stanford, NYU, ParisTech, ENS, …

**TensorFlow**

- Multiple frontends: C++, Python, Java, Go
- Declarative API
- Imperative API through Eager [3]
- Data Flow graphs, with partial graph execution
- CPU, GPU, TPU, Mobile (TensorFlow Lite)
- Google, AirBnb, Uber, SAP, ebay, Intel

Images taken from [1] and [2]
Significance

- Research:
  - Prominent Question: Have you tried using DNN?
- Programming paradigm is shifting
  - Induction, data-centric approach
- TensorFlow and PyTorch are among the primary tools used by the industry and academia
- RNNs are great for data with temporal relations (e.g. text, speech)
Exploration

• Evaluate how the top 2 Deep Learning Frameworks perform in CPU-only computations
  • Lack of available Nvidia GPU :( 
  • and a Google TPU :( 
  • Maybe test on Public Cloud - Amazon Spot Instances? 
• Can you do “Deep” learning on CPUs?
  • Explore the limits on a “commodity” laptop 
  • How far can “fast-prototyping” go?
Working Plan

- So far:
  - Tools installations and playground setup
  - Dataset exploration
  - API familiarisation

- Deliverables:
  - Different RNN depths and architectures comparative benchmarks
  - Accuracy benchmarks
  - Computability benchmarks
  - Results interpretation
Thank you

Q&A

Stefanos Laskaridis
sl829@cam.ac.uk
References

1. PyTorch
   http://pytorch.org/

2. TensorFlow
   https://www.tensorflow.org

3. TensorFlow Eager
   https://research.googleblog.com/2017/10/eager-execution-imperative-define-by.html

4. IMDB Dataset
   http://ai.stanford.edu/~amaas/data/sentiment/
   https://s3.amazonaws.com/text-datasets/imdb.npz
