Python implementation of RNNs

Weiguan Wang
Shanghai University

Overview

1. Various RNN layers
2. Single-layer RNN without intermediate output
3. Multi-layer RNN with sequential output

Various RNN layers

The construction of RNN in Tensorflow follows exactly the same fashion as building DNN, by stacking up layers in a Sequential model.

RNN layers are called from the layers module:

- layers.RNN
- layers.LSTM
- layers.GRU
- ...

Single-layer RNN without intermediate output

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Embedding, Dense

nodes=10
dropout=0.15
input_shape=[10, 40]
model = Sequential()

model.add(LSTM(nodes, dropout=dropout, input_shape=input_shape, use_bias=True))
model.add(Dense(10, activation='relu' , use_bias=True))
model.add(Dense(2, activation ='softmax', use_bias=True))
```

By default, the RNN layer outputs a single vector, corresponding to the output of last time step. This vector is 2-D with shape (batch_size, nodes).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10)                2040

 dense (Dense)               (None, 10)                110

 dense_1 (Dense)             (None, 2)                 22

=================================================================
Total params: 2,172
Trainable params: 2,172
Non-trainable params: 0
_________________________________________________________________
```

Figure: The model summary of the fore-mentioned RNN.

A RNN with sequential output

In some cases, one needs the intermediate outputs of all time steps from the RNN, for example in language models. To this end, one sets ‘return_sequence=True‘.

```
model = Sequential()
model.add(LSTM(nodes, dropout=dropout, input_shape=input_shape, use_bias=True, return_sequences=True))
model.add(LSTM(nodes, dropout=dropout, use_bias=True, return_sequences=True))
model.add(Dense(10, activation='relu' , use_bias=True))
model.add(Dense(2, activation ='softmax', use_bias=True))
```

The following summary shows that the output of the RNN has a shape ‘(batch_size, time_step, nodes)‘

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 10, 10)            2040

 lstm_3 (LSTM)               (None, 10, 10)            840

 dense_4 (Dense)             (None, 10, 10)            110

 dense_5 (Dense)             (None, 10, 2)             22

=================================================================
Total params: 3,012
Trainable params: 3,012
Non-trainable params: 0
```

Figure: The model summary of a RNN with two LSTM layers and sequential output.

Thanks for your attention!