LSTM_NORMALIZATIONS

Xiaohe Xue

#normal_cells_separate:

Other

Form of Normalizations

1. normal_cells

$$
\begin{gathered}
[\,i\ f\ g\ o\,] = norm(h_{t-1}, W_h)\ +\ norm(x_t, W_x) \\
new\_c = (c * \sigma(f + bias) + \sigma(i) * \tanh(j)) \\
(new\_c = batch\_norm(new\_c)\ /\ layer\_norm(new\_c)) \\
new\_h = \tanh(new\_c) * \sigma(o)
\end{gathered}
$$

2. normal_cells_conb

$$
\begin{gathered}
[\,i\ f\ g\ o\,] = norm([\,h_{t-1},\ x_t\,], W) \\
new\_c = (c * \sigma(f + bias) + \sigma(i) * \tanh(j)) \\
(new\_c = batch\_norm(new\_c)\ /\ layer\_norm(new\_c)) \\
new\_h = \tanh(new\_c) * \sigma(o)
\end{gathered}
$$

3. normal_cells_separate

$$
\begin{gathered}
[\,i\ f\ g\ o\,] = [norm(h_{t-1}, w_{ih}),\ norm(h_{t-1}, w_{fh}),\ norm(h_{t-1}, w_{gh}),\ norm(h_{t-1}, w_{oh})] +\ norm(x_t, W_x) \\
new\_c = (c * \sigma(f + bias) + \sigma(i) * \tanh(j)) \\
(new\_c = batch\_norm(new\_c)\ /\ layer\_norm(new\_c)) \\
new\_h = \tanh(new\_c) * \sigma(o)
\end{gathered}
$$

Models

0. General Arguments

```
rnn_mode/cell/model  # cells
                     # base: base model
                     # bn_sep: batch normalization
                     # cn_sep: cosine normalization
                     # ln_sep: layer normalization
                     # wn_sep: weight noramlization
                     # pcc_sep: pcc normalization

lr         #Learning rate
g          #grain, the initial_scale of weight of noramlization
```

1. Sequential MNIST

1.1 Data - MNIST

1.2 Introduction

该项目的每个数据是一个28*28的数字手写体图片。在数据预处理中图片被转换成784*1的向量，所以LSTM一共有784个timestep即LSTM的每次输入都是图片集的单个向量(大小为：batch size * 1)

- 在转换成784*1的向量时，可以尝试固定的随机置换索引方式转换即permuted MNIST。 本项目还未进行尝试。
- 参考
  - https://github.com/OlavHN/bnlstm (有瑕疵)
  - https://gist.github.com/spitis/27ab7d2a30bbaf5ef431b4a02194ac60

1.3 Run

```
python test_784*1.py --cell=base --log_dir=/tmp/logs/ --g=0.5 --lr=0.001
```

1.4 Results

#normal_cells

Rank	Normal	Scale	lr	Accuracy
1	wn	1.0	0.01	0.98203125
2	cn	1.0	0.01	0.97796
3	base	0.0	0.01	0.651671875

#normal_cells_separate

Rank	Normal	Scale	lr	Accuracy
1	wn	1.0	0.01	0.9814609375
2	cn	1.0	0.01	0.9811328125

normal_cells_separate

None

2. PTB

2.1 [Data - PTB]

Download: [simple-examples]

2.2 Introduction

[RNN TensorFlow]

2.3 Run

```
python ptb_word_lm.py --lr=1.0 --g=5.0 --rnn_mode=cn_sep --num_gpus=1 \
--save_path=$HOME/log/ptb_cob
```

2.4 Results - small model

#normal_cells

Rank	Normal	lr	Scale	ppl
1	cn	1.0	5.0	107.70843414
2	pcc	1.0	5.0	108.879889268
3	wn	1.0	1.0	115.074287843
4	base	1.0	0.0	117.544
5	bn	1.0	1.0	129.818630436
6	ln	1.0	1.0	130

#normal_cells_separate

cn 效果依然最好，结果与上述类似

3. DRAW

3.1 Data - MNIST

3.2 Introduction

char-rnn-tensorflow

char-rnn

The Unreasonable Effectiveness of Recurrent Neural Networks

3.3 Run

```
python train.py --model=base --lr=0.001  \
--save_dir=/tmp/char_seq100_refactor/save/base_0.001 \
--log_dir=/tmp/char_seq100_refactor/log/base_0.001
```

3.4 Results

#normal_cells:

Rank	Normal	Scale	lr	cost
1	ln	1.0	0.001	107.849250519
2	wn	1.0	0.001	107.638289787
3	bn	1.0	0.001	111.273218102
4	pcc	5.0	0.001	112.98063578
5	base	0.0	0.01/0.001	120
6	cn	5.0	0.001	123.905787323

#normal_cells_separate:

Rank	Normal	Scale	lr	cost
1	ln	1.0	0.001	104.878376785
2	cn	5.0	0.001	109.033839409
3	wn	1.0	0.001	111.994892326
4	pcc	5.0	0.001	116.779598183

4. NMT

4.1 Data - Neural Machine Translation

- IWSLT'15 English-Vietnamese data **[Small]**
- WMT'14 English-German data **[Medium]**

4.2 Introduction

Neural Machine Translation (seq2seq) Tutorial

4.3 Run

Small Data:

Train: 133K examples, vocab=vocab.(vi|en), train=train.(vi|en) dev=tst2012.(vi|en), test=tst2013.(vi|en), download script.

Training details: We train 2-layer LSTMs of 512 units with bidirectional encoder (i.e., 1 bidirectional layers for the encoder), embedding dim is 512. LuongAttention (scale=True) is used together with dropout keep_prob of 0.8. All parameters are uniformly. We use SGD with learning rate 1.0 as follows: train for 12K steps (~ 12 epochs); after 8K steps, we start halving learning rate every 1K step.

```
python -m nmt.nmt \
    --unit_type=base \
    --src=vi --tgt=en \
    --learning_rate=1.0 \
    --grain=0.0 \
    --vocab_prefix=../../data/nmt_data/vocab  \
    --train_prefix=../../data/nmt_data/train \
    --dev_prefix=../../data/nmt_data/tst2012  \
    --test_prefix=../../data/nmt_data/tst2013 \
    --out_dir=/tmp/nmt_model/base \
    --num_train_steps=12000 \
    --steps_per_stats=100 \
    --num_layers=2 \
    --num_units=128 \
```

```
    --dropout=0.2 \
    --metrics=bleu
```

Medium Data:

Train: 4.5M examples, vocab=vocab.bpe.32000.(de|en),train=train.tok.clean.bpe.32000.(de|en), dev=newstest2013.tok.bpe.32000.(de|en),test=newstest2015.tok.bpe.32000.(de|en),[download script](#)

Training details. Our training hyperparameters are similar to theEnglish-Vietnamese experiments except for the following details. The data issplit into subword units using [BPE](#)(32K operations). We train 4-layer LSTMs of 1024 units with bidirectionalencoder (i.e., 2 bidirectional layers for the encoder), embedding dimis 1024. We train for 350K steps (~ 10 epochs); after 170K steps, we start halving learning rate every 17K step. But, dopout is 0.0, and forget_bias is 0.0.

```
python -m nmt.nmt \
    --unit_type=base \
    --encoder_type=bi \
    --attention=scaled_luong \
    --src=de --tgt=en \
    --vocab_prefix=../../data/nmt_data_large/wmt16_de_en/vocab.bpe.32000  \
    --train_prefix=../../data/nmt_data_large/wmt16_de_en/train.tok.clean.bpe.32000
\
    --dev_prefix=../../data/nmt_data_large/wmt16_de_en/newstest2013.tok.bpe.32000
\
    --test_prefix=../../data/nmt_data_large/wmt16_de_en/newstest2015.tok.bpe.32000
\
    --out_dir=$HOME/log/nmt_attention_model_large\
    --learning_rate=1.0 \
    --grain=1.0 \
    --start_decay_step=170000 \
    --decay_steps=17000 \
    --decay_factor=0.5 \
    --num_train_steps=350000 \
    --steps_per_stats=100 \
    --num_layers=4 \
    --num_units=1024 \
    --dropout=0.0 \
    --forget_bias=0.0 \
    --metrics=bleu
```

4.4 Result

4.4.1 Small Data:

#normal_cells:

Rank	Normal	Scale	lr	bleu test
1	cn	5.0	1.0	6.115/5.719
2	pcc	5.0	1.0	5.838
3	ln	1.0	1.0	5.486
4	wn	1.0	1.0	5.272
5	base	0.0	1.0	4.625/5.177/4.898

#normal_cells_separate:

Rank	Normal	Scale	lr	bleu test
1	cn	5.0	1.0	5.869
2	pcc	5.0	1.0	5.819
3	wn	1.0	1.0	5.559
没有效果	ln	/	/	0.7

4.4.1 Medium Data:

#normal_cells_separate:

Rank	Normal	Scale	lr	bleu test
1	wn	1.0	1.0	30.7
2	pcc	5.0	1.0	30.5
3	cn	5.0	1.0	30.3
4	ln	1.0	1.0	30.2
5	base	0.0	1.0	27.6
6	bn	1.0	1.0	崩溃

Other

1. Cosine Normalization does not well on all vanilla LSTM model
2. Batch Normalization:

- 因为在Batch Normalization在测试时需要用到 population mean 和 population var，所以在Batch Normalization初始化state时，除了 h 和 c ，还有step来标记当前的time step。具体请看代码。
- 因为Batch Normalization的规范化算法是纵向的，所以理论上它在normal_cells上的效果应该和在其余两个cells的效果一样，所以就没有进行试验。

3. Initializer:

- RNN 中 Weights 的initializer 全部等于变量 *weights_initializer*，默认值为None。 生成方式

- Normalization 中scale 变量的initialer全部为*truncated_normal_initializer*。