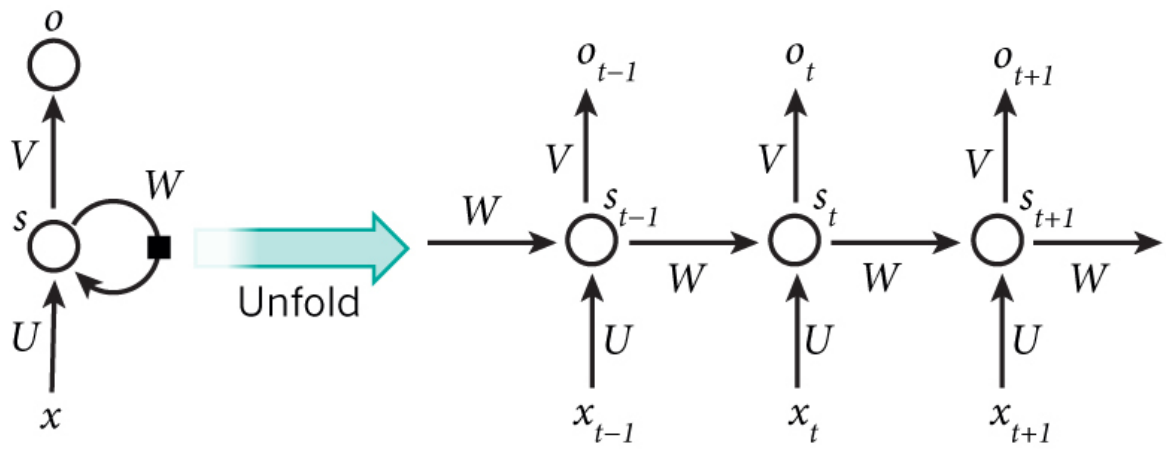
RNN 실습 - 텐서플로우에서 LSTM 및 GRU 사용

1. RNN 모형 및 구성방법
2. 텐서플로우에서 지원하는 RNN 'Cell' 유형
3. 유형별 특성 테스트 위한 코드 구성
4. Test #1 - Vanila RNN
5. Test #2 - Basic LSTM
6. Test #3 - GRU
7. Test #4 - LSTMCell + forget_bias
8. Test #5 - LayerNormBasicLSTMCell
9. Test #6 - LayerNormBasicLSTMCell - what's wrong?
10. 정리

In [1]:
```
%load_ext do_not_print_href
%matplotlib inline
from __future__ import print_function, division
import sys
import time
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

In [2]:
```
!rm -fr logdir
!mkdir -p logdir
```

RNN 모형 (복습)



이미지 출처: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

$$s_t = tanh(Ux_t + Ws_{t-1})$$
$$o_t = softmax(Vs_t)$$

하지만 텐서플로우 RNN 은...

- ***V*** 에 해당하는 구조가 없음
- ***softmax()*** 도 없음
- ***V*** 와 ***softmax()*** 는 필요할 때만 만들어서 붙이면 됨 (`tf.layers.dense`, `tf.nn.softmax`)

RNN 모델 구성 (복습)

- tf.contrib.rnn.BasicRNNCell

```
`__init__`(
  num_units,
  activation=None,
  reuse=None
)
```

- tf.nn.dynamic_rnn

```
dynamic_rnn(
  cell,
  inputs,
  sequence_length=None,
  initial_state=None,
  dtype=None,
  parallel_iterations=None,
  swap_memory=False,
  time_major=False,
  scope=None
)
```

- RNN 구성

```
  cell            = tf.contrib.rnn.BasicRNNCell(
                      num_hidden_units)
  last, states    = tf.nn.dynamic_rnn(
                      cell,
                      inputs,
                      sequence_length=sequence_length,
                      dtype=tf.float32)
```

RNN Cell 종류

- tf.contrib.rnn.BasicRNNCell

```
  ...
```

- tf.contrib.rnn.BasicLSTMCell

```
__init__(
    num_units,
    forget_bias=1.0,
    state_is_tuple=True,
    activation=None,
    reuse=None
)
```

- tf.contrib.rnn.LSTMCell

```
__init__(
    num_units,
    use_peepholes=False,
    cell_clip=None,
    initializer=None,
    num_proj=None,
    proj_clip=None,
    num_unit_shards=None,
    num_proj_shards=None,
    forget_bias=1.0,
    state_is_tuple=True,
    activation=None,
    reuse=None
)
```

- tf.contrib.rnn.GRUCell

```
__init__(
    num_units,
    activation=None,
    reuse=None,
    kernel_initializer=None,
    bias_initializer=None
)
```

- tf.contrib.rnn.LayerNormBasicLSTMCell

> LSTM unit with **layer normalization** and **recurrent dropout.**

```
__init__(
    num_units,
    forget_bias=1.0,
    input_size=None,
    activation=tf.tanh,
    layer_norm=True,
    norm_gain=1.0,
```

```
        norm_shift=0.0,
        dropout_keep_prob=1.0,
        dropout_prob_seed=None,
        reuse=None
    )
```

- 기타 여러가지 Wrapper 지원 - https://www.tensorflow.org/api_guides/python/contrib.rnn

In [3]:
```
from tensorflow.examples.tutorials.mnist.input_data \
    import read_data_sets

mnist = read_data_sets('./mnist', one_hot=False)
```

```
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting ./mnist/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting ./mnist/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting ./mnist/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting ./mnist/t10k-labels-idx1-ubyte.gz
```

In [4]:
```
INPUT_UNITS = 28
NUM_HIDDEN_UNITS = 31

BATCH_SIZE = 128
MAX_SEQ_LEN = 28
```

In [5]:
```
class MnistRnn:
    def __init__(self,
                 inputs,
                 labels,
                 input_units,
                 num_hidden_units,
                 batch_size,
                 max_seq_len,
                 rnn_type,
                 lstm_forget_bias=1.0,
                 add_check = False,
                 lr = 0.001,
                 use_grad_clip = False,
                 ln_norm_smoothing = False):
        '''
        inputs: in shape [batch_size, max_seq_len, input_size]
        labels: in shape [batch_size]
        '''

        # RNN 구성
        if 'basic_rnn' == rnn_type:
            cell = tf.contrib.rnn.BasicRNNCell(num_hidden_units)
        if 'basic_lstm' == rnn_type:
            cell = tf.contrib.rnn.BasicLSTMCell(num_hidden_units)
        if 'lstm' == rnn_type:
            cell = tf.contrib.rnn.LSTMCell(
                        num_hidden_units,
```

```
                forget_bias=lstm_forget_bias)
        if 'gru' == rnn_type:
            cell = tf.contrib.rnn.GRUCell(num_hidden_units)
        if 'ln_basic_lstm' == rnn_type:
            if ln_norm_smoothing:
                cell = tf.contrib.rnn.LayerNormBasicLSTMCell(
                        num_hidden_units,
                        norm_gain=0.85,
                        norm_shift=0.15)
            else:
                cell = tf.contrib.rnn.LayerNormBasicLSTMCell(
                        num_hidden_units)

        sequence_length = [max_seq_len] * batch_size
        last, states    = tf.nn.dynamic_rnn(
                            cell,
                            inputs,
                            sequence_length=sequence_length,
                            dtype=tf.float32)

        # max_seq_len 축으로 0~27 까지 값 중에
        # 0~26 때의 출력 값은 사용하지 않음
        rnn_output = last[:,max_seq_len-1,:]
        # outputs shape will be: [batch_size, 10]
        outputs    = tf.layers.dense(rnn_output, 10)
        loss       = tf.losses.sparse_softmax_cross_entropy(
                        labels, outputs)

        if use_grad_clip:
            tvars_     = tf.trainable_variables()
            grads_, _  = tf.clip_by_global_norm(
                            tf.gradients(
                                loss,
                                tvars_),
                            5.0)
            optimize   = \
                tf.train.AdamOptimizer(learning_rate=lr). \
                            apply_gradients(zip(grads_, tvars_))
        else:
            optimize   = \
                tf.train.AdamOptimizer(learning_rate=lr). \
                            minimize(loss)

        # accuracy
        preds    = tf.argmax(outputs, axis=1)
        errors   = tf.count_nonzero(labels - preds)
        accuracy = 1.0 - tf.cast(errors,tf.float32) / \
                         tf.cast(tf.size(preds),tf.float32)

        # 클래스 객체 외부에서 참고할 수 있도록 속성으로 저장
        self.outputs        = outputs
        self.loss           = loss
        self.optimize       = optimize
        self.accuracy       = accuracy

        # check_numerics
        self.check = [tf.check_numerics(
```

```
                        t,
                        'check_numerics: {}'.format(t.name)) \
                    for t in tf.gradients(
                                loss,
                                tf.trainable_variables()) \
                    if t is not None] \
                   if add_check \
                   else tf.constant(1.0)
```

In [6]:
```
train_loop_count = mnist.train.num_examples // BATCH_SIZE
test_loop_count  = mnist.test.num_examples // BATCH_SIZE

train_loop_count, test_loop_count
```

Out[6]: (429, 78)

In [7]:
```
def train(inputs, labels, max_epochs, train_writer, test_writer):
    step = 0
    for ep in range(max_epochs):

        train_elapsed = []
        train_losses = []
        train_accuracy = []
        for i in range(train_loop_count):
            t_start     = time.time()
            offs        = i * BATCH_SIZE
            batch_input = \
                    mnist.train.images[offs:offs+BATCH_SIZE,:]
            batch_input = \
                    batch_input.reshape(
                            [BATCH_SIZE,
                               MAX_SEQ_LEN,
                               INPUT_UNITS])
            batch_label = \
                    mnist.train.labels[offs:offs+BATCH_SIZE]
            optimize, loss, accuracy, _ = \
                sess.run([model.optimize,
                          model.loss,
                          model.accuracy,
                          model.check],
                         feed_dict = {
                          inputs: batch_input,
                          labels: batch_label })
            train_losses.append(loss)
            train_accuracy.append(accuracy)
            t_elapsed   = time.time() - t_start
            train_elapsed.append(t_elapsed)

            step += 1
            summary = tf.Summary(
                value=[
                    tf.Summary.Value(
                        tag='train_accuracy',
                        simple_value=accuracy
                    ),
                    tf.Summary.Value(
```

```
                    tag='loss',
                    simple_value=loss
                ),
            ]
        )
        train_writer.add_summary(summary,global_step=step)

        if step % 250 == 0:
            print(('[trn] ep {:d}, step {:d}, ' +
                   'loss {:f}, accu {:f}, ' +
                   'sec/iter {:f}').format(
                ep + 1,
                step,
                np.mean(train_losses),
                np.amin(train_accuracy),
                np.mean(train_elapsed)))
            train_losses = []
            train_accuracy = []
            train_elapsed = []

    test_elapsed  = []
    test_accuracy = []
    for i in range(test_loop_count):
        t_start     = time.time()
        offs        = i * BATCH_SIZE
        batch_input = mnist.test.images[offs:offs+BATCH_SIZE,:]
        batch_input = batch_input.reshape(
                        [BATCH_SIZE,
                           MAX_SEQ_LEN,
                           INPUT_UNITS])
        batch_label = mnist.test.labels[offs:offs+BATCH_SIZE]
        accuracy, = \
            sess.run([model.accuracy],
                     feed_dict = {
                      inputs: batch_input,
                      labels: batch_label })
        test_accuracy.append(accuracy)
        t_elapsed   = time.time() - t_start
        test_elapsed.append(t_elapsed)

        step += 1
        summary = tf.Summary(
            value=[
                tf.Summary.Value(
                    tag='test_accuracy',
                    simple_value=accuracy
                ),
            ]
        )
        test_writer.add_summary(summary,global_step=step)

        if step % 250 == 0:
            print(('[tst] ep {:d}, step {:d}, ' +
                   'accu {:f}, sec/iter {:f}').format(
                ep + 1,
                step,
                np.amin(test_accuracy),
```

```
                    np.mean(test_elapsed)))
                test_accuracy = []
                test_elapsed  = []
```

Test #1 - Vanila RNN

In [8]:
```
tf.reset_default_graph()

inputs_ = tf.placeholder(tf.float32,
                         [BATCH_SIZE, MAX_SEQ_LEN, INPUT_UNITS],
                         name='inputs')
labels_ = tf.placeholder(tf.int64,
                         [BATCH_SIZE],
                         name='labels')

model = MnistRnn(inputs_,
                 labels_,
                 INPUT_UNITS,
                 NUM_HIDDEN_UNITS,
                 BATCH_SIZE,
                 MAX_SEQ_LEN,
                 'basic_rnn')
```

In [9]:
```
config = tf.ConfigProto(gpu_options={'allow_growth':True})
sess = tf.InteractiveSession(config=config)

tf.global_variables_initializer().run()

train_writer = tf.summary.FileWriter('logdir/train_basic_rnn',
                                     graph=tf.get_default_graph())
test_writer  = tf.summary.FileWriter('logdir/test_basic_rnn',
                                     graph=tf.get_default_graph())

train(inputs_, labels_, 10, train_writer, test_writer)
```

```
[trn] ep 1, step 250, loss 1.501857, accu 0.085938, sec/iter 0.012
775
[tst] ep 1, step 500, accu 0.500000, sec/iter 0.004678
[trn] ep 2, step 750, loss 0.832589, accu 0.609375, sec/iter 0.011
970
[tst] ep 2, step 1000, accu 0.460938, sec/iter 0.004508
[trn] ep 3, step 1250, loss 0.672037, accu 0.664062, sec/iter 0.01
1896
[tst] ep 3, step 1500, accu 0.695312, sec/iter 0.004475
[trn] ep 4, step 1750, loss 0.583710, accu 0.703125, sec/iter 0.01
1698
[tst] ep 4, step 2000, accu 0.718750, sec/iter 0.004395
[trn] ep 5, step 2250, loss 0.528082, accu 0.710938, sec/iter 0.01
1738
[tst] ep 5, step 2500, accu 0.742188, sec/iter 0.004444
[trn] ep 6, step 2750, loss 0.487420, accu 0.718750, sec/iter 0.01
1778
[tst] ep 6, step 3000, accu 0.765625, sec/iter 0.004677
[trn] ep 7, step 3250, loss 0.454205, accu 0.734375, sec/iter 0.01
1934
[tst] ep 7, step 3500, accu 0.796875, sec/iter 0.004534
[trn] ep 8, step 3750, loss 0.424472, accu 0.765625, sec/iter 0.01
2097
[tst] ep 8, step 4000, accu 0.804688, sec/iter 0.004575
[trn] ep 9, step 4250, loss 0.401004, accu 0.765625, sec/iter 0.01
1978
[tst] ep 9, step 4500, accu 0.804688, sec/iter 0.004578
[trn] ep 10, step 4750, loss 0.385170, accu 0.765625, sec/iter 0.0
12023
[tst] ep 10, step 5000, accu 0.843750, sec/iter 0.004659
```

In [10]:
```
# !tensorboard --ip 0.0.0.0 --logdir logdir
```

Test #2 - Basic LSTM

In [11]:
```
tf.reset_default_graph()

inputs_ = tf.placeholder(tf.float32,
                         [BATCH_SIZE, MAX_SEQ_LEN, INPUT_UNITS],
                         name='inputs')
labels_ = tf.placeholder(tf.int64,
                         [BATCH_SIZE],
                         name='labels')

model = MnistRnn(inputs_,
                 labels_,
                 INPUT_UNITS,
                 NUM_HIDDEN_UNITS,
                 BATCH_SIZE,
                 MAX_SEQ_LEN,
                 'basic_rnn')
```

In [12]:
```
config = tf.ConfigProto(gpu_options={'allow_growth':True})
sess = tf.InteractiveSession(config=config)

tf.global_variables_initializer().run()

train_writer = tf.summary.FileWriter('logdir/train_basic_lstm',
                                     graph=tf.get_default_graph())
test_writer  = tf.summary.FileWriter('logdir/test_basic_lstm',
                                     graph=tf.get_default_graph())

train(inputs_, labels_, 10, train_writer, test_writer)
```

```
[trn] ep 1, step 250, loss 1.556159, accu 0.031250, sec/iter 0.011
665
[tst] ep 1, step 500, accu 0.468750, sec/iter 0.004547
[trn] ep 2, step 750, loss 0.862416, accu 0.539062, sec/iter 0.011
649
[tst] ep 2, step 1000, accu 0.648438, sec/iter 0.004486
[trn] ep 3, step 1250, loss 0.680751, accu 0.609375, sec/iter 0.01
1582
[tst] ep 3, step 1500, accu 0.710938, sec/iter 0.004593
[trn] ep 4, step 1750, loss 0.582057, accu 0.710938, sec/iter 0.01
1579
[tst] ep 4, step 2000, accu 0.750000, sec/iter 0.004539
[trn] ep 5, step 2250, loss 0.511701, accu 0.742188, sec/iter 0.01
1545
[tst] ep 5, step 2500, accu 0.742188, sec/iter 0.004475
[trn] ep 6, step 2750, loss 0.458945, accu 0.765625, sec/iter 0.01
1743
[tst] ep 6, step 3000, accu 0.765625, sec/iter 0.004652
[trn] ep 7, step 3250, loss 0.420503, accu 0.781250, sec/iter 0.01
1834
[tst] ep 7, step 3500, accu 0.796875, sec/iter 0.004464
[trn] ep 8, step 3750, loss 0.382131, accu 0.781250, sec/iter 0.01
2071
[tst] ep 8, step 4000, accu 0.835938, sec/iter 0.004474
[trn] ep 9, step 4250, loss 0.352837, accu 0.796875, sec/iter 0.01
1598
[tst] ep 9, step 4500, accu 0.843750, sec/iter 0.004496
[trn] ep 10, step 4750, loss 0.331150, accu 0.796875, sec/iter 0.0
11719
[tst] ep 10, step 5000, accu 0.875000, sec/iter 0.004524
```

In [13]:
```
# !tensorboard --ip 0.0.0.0 --logdir logdir
```

Test #3 - GRU

In [14]:
```
tf.reset_default_graph()

inputs_ = tf.placeholder(tf.float32,
                         [BATCH_SIZE, MAX_SEQ_LEN, INPUT_UNITS],
                         name='inputs')
labels_ = tf.placeholder(tf.int64,
                         [BATCH_SIZE],
                         name='labels')

model = MnistRnn(inputs_,
                 labels_,
                 INPUT_UNITS,
                 NUM_HIDDEN_UNITS,
                 BATCH_SIZE,
                 MAX_SEQ_LEN,
                 'gru')
```

In [15]:
```
config = tf.ConfigProto(gpu_options={'allow_growth':True})
sess = tf.InteractiveSession(config=config)

tf.global_variables_initializer().run()

train_writer = tf.summary.FileWriter(
                'logdir/train_gru',
                graph=tf.get_default_graph())
test_writer  = tf.summary.FileWriter(
                'logdir/test_gru',
                graph=tf.get_default_graph())

train(inputs_, labels_, 10, train_writer, test_writer)
```

```
[trn] ep 1, step 250, loss 1.510569, accu 0.039062, sec/iter 0.025
047
[tst] ep 1, step 500, accu 0.609375, sec/iter 0.007069
[trn] ep 2, step 750, loss 0.485274, accu 0.632812, sec/iter 0.024
859
[tst] ep 2, step 1000, accu 0.812500, sec/iter 0.006946
[trn] ep 3, step 1250, loss 0.292094, accu 0.804688, sec/iter 0.02
4922
[tst] ep 3, step 1500, accu 0.828125, sec/iter 0.007017
[trn] ep 4, step 1750, loss 0.226324, accu 0.843750, sec/iter 0.02
4893
[tst] ep 4, step 2000, accu 0.890625, sec/iter 0.006976
[trn] ep 5, step 2250, loss 0.191989, accu 0.882812, sec/iter 0.02
4956
[tst] ep 5, step 2500, accu 0.890625, sec/iter 0.007160
[trn] ep 6, step 2750, loss 0.167970, accu 0.890625, sec/iter 0.02
4914
[tst] ep 6, step 3000, accu 0.898438, sec/iter 0.007287
[trn] ep 7, step 3250, loss 0.147263, accu 0.898438, sec/iter 0.02
4811
[tst] ep 7, step 3500, accu 0.890625, sec/iter 0.006978
[trn] ep 8, step 3750, loss 0.130540, accu 0.906250, sec/iter 0.02
5367
[tst] ep 8, step 4000, accu 0.914062, sec/iter 0.006976
[trn] ep 9, step 4250, loss 0.117612, accu 0.914062, sec/iter 0.02
4737
[tst] ep 9, step 4500, accu 0.929688, sec/iter 0.006923
[trn] ep 10, step 4750, loss 0.107138, accu 0.914062, sec/iter 0.0
24723
[tst] ep 10, step 5000, accu 0.929688, sec/iter 0.006900
```

In [16]:
```
# !tensorboard --ip 0.0.0.0 --logdir logdir
```

Test #4 - LSTMCell + forget_bias

In [17]:
```
tf.reset_default_graph()

inputs_ = tf.placeholder(tf.float32,
                         [BATCH_SIZE, MAX_SEQ_LEN, INPUT_UNITS],
                         name='inputs')
labels_ = tf.placeholder(tf.int64,
                         [BATCH_SIZE],
                         name='labels')

model = MnistRnn(inputs_,
                 labels_,
                 INPUT_UNITS,
                 NUM_HIDDEN_UNITS,
                 BATCH_SIZE,
                 MAX_SEQ_LEN,
                 'lstm')
```

In [18]:
```
config = tf.ConfigProto(gpu_options={'allow_growth':True})
sess = tf.InteractiveSession(config=config)

tf.global_variables_initializer().run()

train_writer = tf.summary.FileWriter('logdir/train_lstm',
                                     graph=tf.get_default_graph())
test_writer  = tf.summary.FileWriter('logdir/test_lstm',
                                     graph=tf.get_default_graph())
```

In [19]:
```
train(inputs_, labels_, 10, train_writer, test_writer)
```

```
[trn] ep 1, step 250, loss 1.497667, accu 0.140625, sec/iter 0.025
244
[tst] ep 1, step 500, accu 0.656250, sec/iter 0.006836
[trn] ep 2, step 750, loss 0.511671, accu 0.656250, sec/iter 0.025
077
[tst] ep 2, step 1000, accu 0.796875, sec/iter 0.006647
[trn] ep 3, step 1250, loss 0.329038, accu 0.773438, sec/iter 0.02
5015
[tst] ep 3, step 1500, accu 0.820312, sec/iter 0.006692
[trn] ep 4, step 1750, loss 0.245046, accu 0.828125, sec/iter 0.02
4962
[tst] ep 4, step 2000, accu 0.851562, sec/iter 0.006821
[trn] ep 5, step 2250, loss 0.200933, accu 0.859375, sec/iter 0.02
5588
[tst] ep 5, step 2500, accu 0.859375, sec/iter 0.006696
[trn] ep 6, step 2750, loss 0.172842, accu 0.875000, sec/iter 0.02
6285
[tst] ep 6, step 3000, accu 0.890625, sec/iter 0.006996
[trn] ep 7, step 3250, loss 0.151646, accu 0.882812, sec/iter 0.02
5638
[tst] ep 7, step 3500, accu 0.906250, sec/iter 0.006747
[trn] ep 8, step 3750, loss 0.135656, accu 0.898438, sec/iter 0.02
5064
[tst] ep 8, step 4000, accu 0.906250, sec/iter 0.006665
[trn] ep 9, step 4250, loss 0.123542, accu 0.898438, sec/iter 0.02
5279
[tst] ep 9, step 4500, accu 0.914062, sec/iter 0.006632
[trn] ep 10, step 4750, loss 0.113314, accu 0.914062, sec/iter 0.0
25696
[tst] ep 10, step 5000, accu 0.945312, sec/iter 0.006685
```

In [20]:
```
# !tensorboard --ip 0.0.0.0 --logdir logdir
```

Test #5 - LayerNormBasicLSTMCell

In [21]:
```
tf.reset_default_graph()

inputs_ = tf.placeholder(tf.float32,
                         [BATCH_SIZE, MAX_SEQ_LEN, INPUT_UNITS],
                         name='inputs')
labels_ = tf.placeholder(tf.int64,
                         [BATCH_SIZE],
                         name='labels')

model = MnistRnn(inputs_,
                 labels_,
                 INPUT_UNITS,
                 NUM_HIDDEN_UNITS,
                 BATCH_SIZE,
                 MAX_SEQ_LEN,
                 'ln_basic_lstm')
```

In [23]:
```
config = tf.ConfigProto(gpu_options={'allow_growth':True})
sess = tf.InteractiveSession(config=config)

tf.global_variables_initializer().run()

train_writer = tf.summary.FileWriter('logdir/train_ln_basic_lstm',
                                     graph=tf.get_default_graph())
test_writer  = tf.summary.FileWriter('logdir/test_ln_basic_lstm',
                                     graph=tf.get_default_graph())

train(inputs_, labels_, 1, train_writer, test_writer)
```

```
[trn] ep 1, step 250, loss nan, accu 0.039062, sec/iter 0.113048
[tst] ep 1, step 500, accu 0.054688, sec/iter 0.034684
```

In [24]:
```
# !tensorboard --ip 0.0.0.0 --logdir logdir
```

Test #6 - LayerNormBasicLSTMCell - what's wrong?

- tf.check_numerics()

  > When run, reports an `InvalidArgument` error if tensor has any values that are not a number (NaN) or infinity (Inf). Otherwise, passes tensor as-is.

```
check_numerics(
    tensor,
    message,
    name=None
)
```

- tf.gradients()

  > Constructs symbolic partial derivatives of sum of `ys` w.r.t. x in `xs`

```
gradients(
    ys,
    xs,
    grad_ys=None,
    name='gradients',
    colocate_gradients_with_ops=False,
    gate_gradients=False,
    aggregation_method=None
)
```

- 사용 예:

```
# check_numerics
self.check = [tf.check_numerics(t,
                'check_numerics: {}'.format(t.name)) \
              for t in tf.gradients(
                          loss,
                          tf.trainable_variables()) \
              if t is not None]
...
summary, optimize, loss, accuracy, _ = \
    sess.run([model.train_summary,
              model.optimize,
              model.loss,
              model.accuracy,
              model.check],
              ...
```

In [25]:
```
tf.reset_default_graph()

inputs_ = tf.placeholder(tf.float32,
                         [BATCH_SIZE, MAX_SEQ_LEN, INPUT_UNITS],
                         name='inputs')
labels_ = tf.placeholder(tf.int64,
                         [BATCH_SIZE],
                         name='labels')

model = MnistRnn(inputs_,
                 labels_,
                 INPUT_UNITS,
                 NUM_HIDDEN_UNITS,
                 BATCH_SIZE,
                 MAX_SEQ_LEN,
                 'ln_basic_lstm',
                 add_check = True)

# 여기서 NaN 문제가 생기는 걸 확인했으면, 다음 방법들을 시도 해 보세요
#  1. learning_rate 를 줄여
#  2. 적당한 값으로 gradient clipping
#  3. 사용한 컴퍼넌트에 smoothing 할 수 있는 파라메터가 있는지 확인하고 적용

#         lr = 0.0001
#         use_grad_clip = Trues
#         ln_norm_smoothing = True
```

In [ ]:
```
config = tf.ConfigProto(gpu_options={'allow_growth':True})
sess = tf.InteractiveSession(config=config)

tf.global_variables_initializer().run()

train_writer = tf.summary.FileWriter(
                'logdir/train_ln_basic_lstm_2',
                graph=tf.get_default_graph())
test_writer  = tf.summary.FileWriter(
                'logdir/test_ln_basic_lstm_2',
                graph=tf.get_default_graph())

train(inputs_, labels_, 10, train_writer, test_writer)
```

In [27]:
```
# !tensorboard --ip 0.0.0.0 --logdir logdir
```

정리해 봅시다

텐서플로우에서 지원하는 RNN Cell 유형

- tf.contrib.rnn.BasicRNNCell()
- tf.contrib.rnn.BasicLSTMCell()
- tf.contrib.rnn.LSTMCell()
- tf.contrib.rnn.GRUCell()
- tf.contrib.rnn.LayerNormBasicLSTMCell()

RNN 구성

- tf.nn.dynamic_rnn()

```
cell         = tf.contrib.rnn.BasicRNNCell(
                  num_hidden_units)
last, states = tf.nn.dynamic_rnn(
                     cell,
                     inputs,
                     sequence_length=sequence_length,
                     dtype=tf.float32)
```

실행중 NaN, Inf 등이 발생하면?

1. tf.check_numerics()로 문제가 되는 컴퍼넌트 파악
2. learning_rate 를 줄여본다
3. 적당한 값으로 gradient clipping 해 본다
4. 사용한 컴퍼넌트에 smoothing 할 수 있는 파라메터가 있는지 확인하고 적용해 본다

gradient clipping?

- tf.clip_by_global_norm

```
clip_by_global_norm(
    t_list,
    clip_norm,
    use_norm=None,
    name=None
)
```

- tf.gradients

```
gradients(
    ys,
    xs,
    grad_ys=None,
    name='gradients',
    colocate_gradients_with_ops=False,
    gate_gradients=False,
    aggregation_method=None
)
```

- optimizer.apply_gradients

```
apply_gradients(
    grads_and_vars,
    global_step=None,
    name=None
)
```

- 사용예:

```
tvars_     = tf.trainable_variables()
grads_, _  = tf.clip_by_global_norm(
                tf.gradients(loss,tvars_),
                5.0)
optimize   = tf.train.AdamOptimizer(
                learning_rate=learning_rate) \
                .apply_gradients(zip(grads_, tvars_))
```