2_pam_model

May 12, 2019

0.1 Importing Required Packages

[0]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.layers import *
from sklearn import preprocessing
import tensorflow.keras.backend as K
from sklearn.metrics import mean_squared_error
```

0.2 Hyper Parameters

[0]:

```
msg_total = 2
channel = 4
epochs = 5000
sigma = 1e-4
batch_size = 1024
```

0.3 Defiing required functions

[0]:

```
def perturbation(x):
    w = K.random_normal(shape = (channel,2), mean=0.0,stddev=sigma**0.
 ↪5,dtype=None)
    xp = ((1-sigma)**0.5)*x + w
    return xp

def loss_tx(y_true, y_pred):
    return -y_true*y_pred

def get_policy(inp):
    xp = inp[0]
    x = inp[1]
    w = xp - x
    policy = -K.sum(w*w)
    return policy
```

1 Transmitter

1.1 Defining Architecture

[0]:
```
tx_inp = Input((1,))
embbedings_layer = Dense(msg_total, activation = 'relu')(tx_inp)
layer_dense = Dense(2*channel, activation = 'relu')(embbedings_layer)
to_complex = Reshape((channel,2))(layer_dense)
x = Lambda(lambda x: keras.backend.l2_normalize(x))(to_complex)
xp = Lambda(perturbation)(to_complex)
policy = Lambda(get_policy)([xp,x])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/resource_variable_ops.py:435: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
```

1.2 Declaring Models

[0]:
```
model_policy = keras.models.Model(inputs=tx_inp, outputs=policy)
model_tx = keras.models.Model(inputs=tx_inp, outputs=xp)
model_x = keras.models.Model(inputs=tx_inp, outputs=x)

model_policy.compile(loss=loss_tx, optimizer=tf.keras.optimizers.SGD(lr = 1e-5))
print(model_policy.summary())
```

```
--------------------------------------------------------------------------------
------------------
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            (None, 1)            0

--------------------------------------------------------------------------------
------------------
dense (Dense)                   (None, 2)            4           input_1[0][0]
--------------------------------------------------------------------------------
------------------
dense_1 (Dense)                 (None, 8)            24          dense[0][0]
--------------------------------------------------------------------------------
------------------
reshape (Reshape)               (None, 4, 2)         0           dense_1[0][0]
--------------------------------------------------------------------------------
```

```
------------------
lambda_1 (Lambda)               (None, 4, 2)         0           reshape[0][0]

--------------------------------------------------------------------------------
------------------
lambda (Lambda)                 (None, 4, 2)         0           reshape[0][0]

--------------------------------------------------------------------------------
------------------
lambda_2 (Lambda)               ()                   0           lambda_1[0][0]
                                                                 lambda[0][0]
================================================================================
==================
Total params: 28
Trainable params: 28
Non-trainable params: 0
--------------------------------------------------------------------------------
------------------
None
```

2 Receiver

2.1 Defining Architecture

[0]:

```
rx_inp = Input((channel,2))
to_flat = Reshape((2*channel,))(rx_inp)
fc = Dense(5*2*channel, activation = 'relu')(to_flat)
softmax = Dense(msg_total, activation = 'softmax')(fc)

model_rx = keras.models.Model(inputs=rx_inp, outputs=softmax)

model_rx.compile(loss=tf.keras.losses.categorical_crossentropy, optimizer=tf.
  ↪keras.optimizers.Adam())
print(model_rx.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 4, 2)              0

_________________________________________________________________
reshape_1 (Reshape)          (None, 8)                 0

_________________________________________________________________
dense_2 (Dense)              (None, 40)                360

_________________________________________________________________
dense_3 (Dense)              (None, 2)                 82
=================================================================
Total params: 442
Trainable params: 442
Non-trainable params: 0
```

```
-----------------------------------------------------------------
None
```

2.2 Alternative Training

[0]:
```
loss_tx = []
loss_rx = []
for epoch in range(epochs):
#     Transmitter training
    raw_input = np.random.randint(0,msg_total,(batch_size))
    label = np.zeros((batch_size, msg_total))
    label[np.arange(batch_size), raw_input] = 1
    tx_input = raw_input/float(msg_total)
    xp = model_tx.predict(tx_input)
    y = xp + np.random.normal(0,0.001,(batch_size, channel,2))
    pred = model_rx.predict(y)
    loss = np.sum(np.square(label - pred), axis = 1)
    history_tx = model_policy.fit(tx_input, loss, batch_size=batch_size,␣
 ↪epochs=1, verbose=0)
    loss_tx.append(history_tx.history['loss'][0])

#     Receiver Training
    raw_input = np.random.randint(0,msg_total,(batch_size))
    label = np.zeros((batch_size, msg_total))
    label[np.arange(batch_size), raw_input] = 1
    tx_input = raw_input/float(msg_total)
    x = model_x.predict(tx_input)
    y = x + np.random.normal(0,0.001,(batch_size, channel,2))
    history_rx = model_rx.fit(y, label, batch_size=batch_size, epochs=1,␣
 ↪verbose=0)
    loss_rx.append(history_rx.history['loss'][0])

    if(epoch % 100 == 0):
        print('epoch: ', epoch, 'tx_loss', history_tx.history['loss'][0],␣
 ↪'rx_loss', history_rx.history['loss'][0])
```

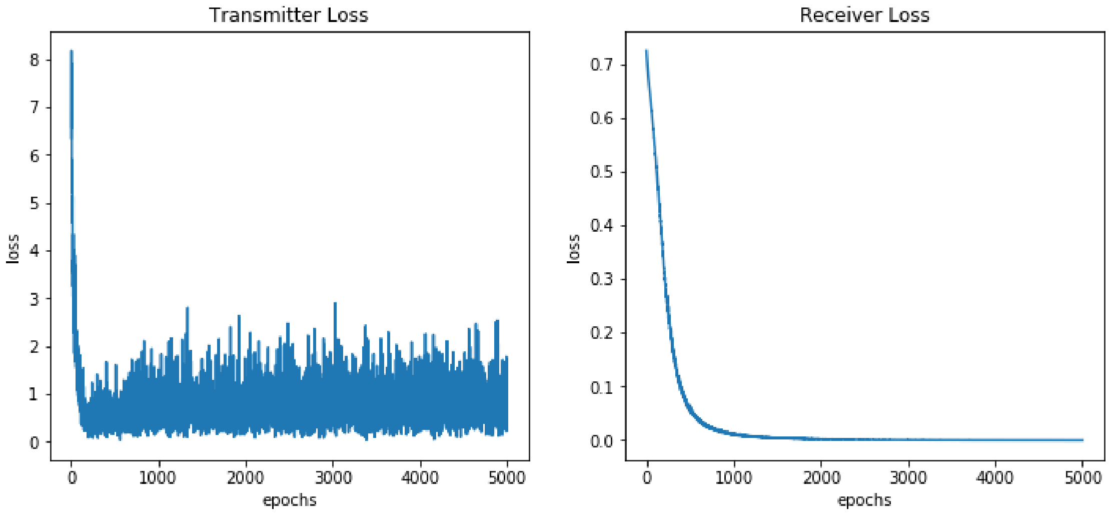
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
epoch:  0 tx_loss 6.355036735534668 rx_loss 0.7242905497550964
epoch:  100 tx_loss 1.8137860298156738 rx_loss 0.5351258516311646
epoch:  200 tx_loss 0.8131426572799683 rx_loss 0.31978639960289
epoch:  300 tx_loss 0.4941194951534271 rx_loss 0.1713171899318695
epoch:  400 tx_loss 0.3641790747642517 rx_loss 0.09444549679756165
```

```
epoch:  500 tx_loss 0.7153945565223694 rx_loss 0.054864201694726944
epoch:  600 tx_loss 0.4647824764251709 rx_loss 0.04075027257204056
epoch:  700 tx_loss 0.4751901626586914 rx_loss 0.025828994810581207
epoch:  800 tx_loss 0.5345849990844727 rx_loss 0.02043948322534561
epoch:  900 tx_loss 0.3673986792564392 rx_loss 0.014428166672587395
epoch:  1000 tx_loss 0.3670782148838043 rx_loss 0.011922061443328857
epoch:  1100 tx_loss 0.9511679410934448 rx_loss 0.00840630941092968
epoch:  1200 tx_loss 0.6923137903213501 rx_loss 0.007425171323120594
epoch:  1300 tx_loss 0.741775393486023 rx_loss 0.00629937369376421
epoch:  1400 tx_loss 0.9444401860237122 rx_loss 0.005360973533242941
epoch:  1500 tx_loss 1.7412992715835571 rx_loss 0.004391422029584646
epoch:  1600 tx_loss 0.6625553369522095 rx_loss 0.00392648670822382
epoch:  1700 tx_loss 0.8560501933097839 rx_loss 0.0038145408034324646
epoch:  1800 tx_loss 0.7468227744102478 rx_loss 0.0030808132141828537
epoch:  1900 tx_loss 0.7250442504882812 rx_loss 0.002631203504279256
epoch:  2000 tx_loss 0.528317928314209 rx_loss 0.002227142918854952
epoch:  2100 tx_loss 0.42298823595046997 rx_loss 0.0019248031312599778
epoch:  2200 tx_loss 0.3724147081375122 rx_loss 0.0018200164195150137
epoch:  2300 tx_loss 0.598807692527771 rx_loss 0.0014720705803483725
epoch:  2400 tx_loss 0.570252537727356 rx_loss 0.0015578294405713677
epoch:  2500 tx_loss 1.3871800899505615 rx_loss 0.0013590501621365547
epoch:  2600 tx_loss 0.4627700746059418 rx_loss 0.001310709398239851
epoch:  2700 tx_loss 1.3259549140930176 rx_loss 0.001097007654607296
epoch:  2800 tx_loss 1.886702537536621 rx_loss 0.0010604213457554579
epoch:  2900 tx_loss 1.357552409172058 rx_loss 0.0008800771902315319
epoch:  3000 tx_loss 0.5879579782485962 rx_loss 0.0009826362365856767
epoch:  3100 tx_loss 1.249929666519165 rx_loss 0.0008248339290730655
epoch:  3200 tx_loss 0.49422529339790344 rx_loss 0.0007180306711234152
epoch:  3300 tx_loss 0.7422264814376831 rx_loss 0.0005964445881545544
epoch:  3400 tx_loss 1.3139021396636963 rx_loss 0.0006083508487790823
epoch:  3500 tx_loss 0.5861632823944092 rx_loss 0.0004838921595364809
epoch:  3600 tx_loss 0.1856534779071808 rx_loss 0.0006139599718153477
epoch:  3700 tx_loss 0.494204044342041 rx_loss 0.00048443354899063706
epoch:  3800 tx_loss 0.6790932416915894 rx_loss 0.0004982298123650253
epoch:  3900 tx_loss 0.8209909200668335 rx_loss 0.0003675811749417335
epoch:  4000 tx_loss 0.7746607065200806 rx_loss 0.00046498404117301106
epoch:  4100 tx_loss 0.8726150989532471 rx_loss 0.00035181891871616244
epoch:  4200 tx_loss 0.4192732572555542 rx_loss 0.0003383115981705487
epoch:  4300 tx_loss 1.2453705072402954 rx_loss 0.0003893995308317244
epoch:  4400 tx_loss 1.0274486541748047 rx_loss 0.0003541375626809895
epoch:  4500 tx_loss 0.4736355245113373 rx_loss 0.00022588712454307824
epoch:  4600 tx_loss 0.7548935413360596 rx_loss 0.0003139497130177915
epoch:  4700 tx_loss 0.8765056133270264 rx_loss 0.0002393309841863811
epoch:  4800 tx_loss 0.9064711332321167 rx_loss 0.00023339706240221858
epoch:  4900 tx_loss 0.5094815492630005 rx_loss 0.00019088402041234076
```

2.3 Plotting Transmitter and Receiver Losses

[0]:

```python
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
plt.plot(loss_tx)
plt.title('Transmitter Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.subplot(1,2,2)
plt.plot(loss_rx)
plt.title('Receiver Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.show()
```



2.4 Prediction

[0]:

```python
#testing
batch_size = 100
raw_input = np.random.randint(0,msg_total,(batch_size))
print(raw_input)
label = np.zeros((batch_size, msg_total))
label[np.arange(batch_size), raw_input] = 1
tx_input = raw_input/float(msg_total)
xp = model_x.predict(tx_input)
y = xp + np.random.normal(0,0.001,(batch_size, channel,2))
pred = model_rx.predict(y)
pred_int = np.argmax(pred, axis = 1)
print(pred_int)
```

```
from sklearn.metrics import accuracy_score

print('accuracy:',accuracy_score(raw_input, pred_int))
```

```
[1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0
 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 1 1 0 0 0
 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 1]
[1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0
 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 1 1 0 0 0
 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 1]
accuracy: 1.0
```

[0]:

```
```