TP02-LinearRegression

September 18, 2019

1 Linear Regression

Observe and run the following code. It performs a linear regression over an artificial dataset.

[1]:
```
import numpy as np
import tensorflow as tf


def generate_all_dataset():
    # --- Fake dataset ---

    np.random.seed(0)

    ntrain = 1000
    nvalid = 100
    ntest = 100

    # number of features
    nc = 4
    # number of outputs
    no = 2

    A = np.random.randn(nc, no)
    b = np.random.randn(no)

    print("True A\n", A)
    print("True b\n", b)

    def generate_one_dataset(A, b, n):
        nc, no = A.shape
        X = np.random.randn(n, nc)
        Y = X.dot(A)+b
        return np.array(X, dtype='float32'), np.array(Y, dtype='float32')

    Xtrain, Ytrain = generate_one_dataset(A, b, ntrain)
    Xvalid, Yvalid = generate_one_dataset(A, b, nvalid)
    Xtest, Ytest = generate_one_dataset(A, b, ntest)
```

```
    return (Xtrain, Ytrain), (Xvalid, Yvalid), (Xtest, Ytest)


def linear_regression_v1(trainset, validset, testset):

    Xtrain, Ytrain = trainset
    Xvalid, Yvalid = validset
    Xtest, Ytest = testset

    # --- Linear Regression ---

    # NEW ! reset tensorflow
    tf.reset_default_graph()

    # Hyperparameters
    learning_rate = 5e-2
    training_epochs = 100

    # Xtrain (nxtrain,nc)
    nxtrain, nc = Xtrain.shape
    nxtest, _ = Xtest.shape
    # Ytrain (nxtrain,no)
    no = Ytrain.shape[1]

    # Placeholders for exchanging between computer memory
    # and tensorflow compute engine.
    # We put None on teh shape where the value is unkown
    # (here the number of exemples)
    x = tf.placeholder(tf.float32, [None, nc])
    y = tf.placeholder(tf.float32, [None, no])

    # Linear regression parameters
    A = tf.Variable(tf.zeros([nc, no]))
    b = tf.Variable(tf.zeros([no]))
    # All parameters are gathered into var_list
    var_list = [A, b]

    # Actual linear regression
    pred = tf.matmul(x, A) + b

    # Model loss
    loss = tf.losses.mean_squared_error(y, pred)

    # NEW ! Optimizer based on gradient descent
    # It computes gradients of loss over var_list items
    # and construct the updates for var_list items.
```

```
    # We don't have to write the gradient step by ourselves
    optimizer = tf.train.GradientDescentOptimizer(
        learning_rate).minimize(loss, var_list=var_list)

    # NEW ! Initializer for all the variables
    # Equivalent to A.initializer + b.initializer
    init_global = tf.initializers.global_variables()

    # NEW ! All the operations will be run in a unique
    # session that will be closed at the end of the calculus
    with tf.Session() as session:

        # We call the initialization of A and b
        session.run(init_global)

        # We compute the train and validation loss
        # Note that you just have to change the feed_dict to change the set
        trainloss, = session.run([loss], feed_dict={x: Xtrain, y: Ytrain})
        validloss, = session.run([loss], feed_dict={x: Xvalid, y: Yvalid})
        print("Init\t\t train loss %f\t valid loss %f" %
              (trainloss, validloss))

        # We cycle on epochs
        for epoch in range(training_epochs):
            # We do the gradient step and compute the train loss
            _, trainloss, = session.run(
                [optimizer, loss], feed_dict={x: Xtrain, y: Ytrain})
            # We compute the valid loss
            validloss, = session.run([loss], feed_dict={x: Xvalid, y: Yvalid})

            print("Epoch %03d\t train loss %f\t valid loss %f" %
                  (epoch, trainloss, validloss))

        # We compute the test loss
        testloss, = session.run([loss], feed_dict={x: Xtest, y: Ytest})

        print("Test loss %f" % (testloss,))

        # Found parameters
        Aval, bval = session.run(var_list)
        print("Estimated A\n", Aval)
        print('Estimated b\n', bval)
    # Here session is closed automatically


linear_regression_v1(*generate_all_dataset())
```

```
True A
 [[ 1.76405235  0.40015721]
 [ 0.97873798  2.2408932 ]
 [ 1.86755799 -0.97727788]
 [ 0.95008842 -0.15135721]]
True b
 [-0.10321885  0.4105985 ]
WARNING:tensorflow:From /home/rherault/.local/venvs/spyder/lib/python3.7/site-
packages/tensorflow/python/framework/op_def_library.py:263: 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.
WARNING:tensorflow:From /home/rherault/.local/venvs/spyder/lib/python3.7/site-
packages/tensorflow/python/ops/losses/losses_impl.py:667: to_float (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Init             train loss 7.184080     valid loss 7.740232
Epoch 000        train loss 7.184080     valid loss 7.006136
Epoch 001        train loss 6.501032     valid loss 6.341899
Epoch 002        train loss 5.883163     valid loss 5.740851
Epoch 003        train loss 5.324234     valid loss 5.196962
Epoch 004        train loss 4.818598     valid loss 4.704776
Epoch 005        train loss 4.361157     valid loss 4.259362
Epoch 006        train loss 3.947302     valid loss 3.856260
Epoch 007        train loss 3.572863     valid loss 3.491436
Epoch 008        train loss 3.234073     valid loss 3.161245
Epoch 009        train loss 2.927526     valid loss 2.862387
Epoch 010        train loss 2.650142     valid loss 2.591878
Epoch 011        train loss 2.399136     valid loss 2.347021
Epoch 012        train loss 2.171991     valid loss 2.125374
Epoch 013        train loss 1.966431     valid loss 1.924729
Epoch 014        train loss 1.780396     valid loss 1.743091
Epoch 015        train loss 1.612025     valid loss 1.578652
Epoch 016        train loss 1.459636     valid loss 1.429779
Epoch 017        train loss 1.321705     valid loss 1.294993
Epoch 018        train loss 1.196856     valid loss 1.172956
Epoch 019        train loss 1.083843     valid loss 1.062458
Epoch 020        train loss 0.981541     valid loss 0.962405
Epoch 021        train loss 0.888930     valid loss 0.871806
Epoch 022        train loss 0.805089     valid loss 0.789764
Epoch 023        train loss 0.729184     valid loss 0.715469
Epoch 024        train loss 0.660462     valid loss 0.648187
Epoch 025        train loss 0.598241     valid loss 0.587253
Epoch 026        train loss 0.541902     valid loss 0.532067
Epoch 027        train loss 0.490889     valid loss 0.482084
```

```
Epoch 028       train loss 0.444695     valid loss 0.436812
Epoch 029       train loss 0.402864     valid loss 0.395806
Epoch 030       train loss 0.364982     valid loss 0.358663
Epoch 031       train loss 0.330676     valid loss 0.325016
Epoch 032       train loss 0.299605     valid loss 0.294537
Epoch 033       train loss 0.271465     valid loss 0.266925
Epoch 034       train loss 0.245977     valid loss 0.241911
Epoch 035       train loss 0.222891     valid loss 0.219248
Epoch 036       train loss 0.201979     valid loss 0.198716
Epoch 037       train loss 0.183037     valid loss 0.180112
Epoch 038       train loss 0.165877     valid loss 0.163257
Epoch 039       train loss 0.150332     valid loss 0.147983
Epoch 040       train loss 0.136249     valid loss 0.134144
Epoch 041       train loss 0.123490     valid loss 0.121603
Epoch 042       train loss 0.111930     valid loss 0.110238
Epoch 043       train loss 0.101456     valid loss 0.099939
Epoch 044       train loss 0.091966     valid loss 0.090605
Epoch 045       train loss 0.083366     valid loss 0.082146
Epoch 046       train loss 0.075574     valid loss 0.074479
Epoch 047       train loss 0.068512     valid loss 0.067530
Epoch 048       train loss 0.062113     valid loss 0.061232
Epoch 049       train loss 0.056314     valid loss 0.055522
Epoch 050       train loss 0.051058     valid loss 0.050347
Epoch 051       train loss 0.046294     valid loss 0.045656
Epoch 052       train loss 0.041976     valid loss 0.041403
Epoch 053       train loss 0.038062     valid loss 0.037548
Epoch 054       train loss 0.034515     valid loss 0.034053
Epoch 055       train loss 0.031299     valid loss 0.030884
Epoch 056       train loss 0.028384     valid loss 0.028011
Epoch 057       train loss 0.025742     valid loss 0.025406
Epoch 058       train loss 0.023346     valid loss 0.023045
Epoch 059       train loss 0.021174     valid loss 0.020903
Epoch 060       train loss 0.019205     valid loss 0.018961
Epoch 061       train loss 0.017419     valid loss 0.017200
Epoch 062       train loss 0.015800     valid loss 0.015603
Epoch 063       train loss 0.014333     valid loss 0.014155
Epoch 064       train loss 0.013001     valid loss 0.012842
Epoch 065       train loss 0.011794     valid loss 0.011651
Epoch 066       train loss 0.010700     valid loss 0.010570
Epoch 067       train loss 0.009707     valid loss 0.009591
Epoch 068       train loss 0.008807     valid loss 0.008702
Epoch 069       train loss 0.007990     valid loss 0.007896
Epoch 070       train loss 0.007250     valid loss 0.007165
Epoch 071       train loss 0.006578     valid loss 0.006501
Epoch 072       train loss 0.005969     valid loss 0.005900
Epoch 073       train loss 0.005416     valid loss 0.005354
Epoch 074       train loss 0.004915     valid loss 0.004859
Epoch 075       train loss 0.004460     valid loss 0.004409
```

```
Epoch 076        train loss 0.004048     valid loss 0.004002
Epoch 077        train loss 0.003674     valid loss 0.003632
Epoch 078        train loss 0.003334     valid loss 0.003296
Epoch 079        train loss 0.003026     valid loss 0.002992
Epoch 080        train loss 0.002746     valid loss 0.002716
Epoch 081        train loss 0.002493     valid loss 0.002465
Epoch 082        train loss 0.002263     valid loss 0.002238
Epoch 083        train loss 0.002054     valid loss 0.002031
Epoch 084        train loss 0.001864     valid loss 0.001844
Epoch 085        train loss 0.001693     valid loss 0.001674
Epoch 086        train loss 0.001537     valid loss 0.001520
Epoch 087        train loss 0.001395     valid loss 0.001380
Epoch 088        train loss 0.001266     valid loss 0.001253
Epoch 089        train loss 0.001150     valid loss 0.001137
Epoch 090        train loss 0.001044     valid loss 0.001033
Epoch 091        train loss 0.000948     valid loss 0.000938
Epoch 092        train loss 0.000861     valid loss 0.000851
Epoch 093        train loss 0.000781     valid loss 0.000773
Epoch 094        train loss 0.000710     valid loss 0.000702
Epoch 095        train loss 0.000644     valid loss 0.000637
Epoch 096        train loss 0.000585     valid loss 0.000579
Epoch 097        train loss 0.000531     valid loss 0.000526
Epoch 098        train loss 0.000483     valid loss 0.000477
Epoch 099        train loss 0.000438     valid loss 0.000434
Test loss 0.000440
Estimated A
 [[ 1.7502961   0.39727157]
 [ 0.9725292   2.2264218 ]
 [ 1.8526078  -0.97195363]
 [ 0.94003755 -0.14778528]]
Estimated b
 [-0.10920743  0.40893131]
```

1.1 Dataset

In the precedent example, all the dataset is used at each gradient step. In most of the case, you can't do that as it will not fit into the memory of the computation engine. You have to split your data into little batches. Tensorflow dataset are here to help you in that process.

Let's observe the next script. It presents how to create a Tensorflow dataset object and how to create bacthes to iterate over it. For each batch, the content of the bacth is displayed as well as the computation of a phony operation.

[2]:
```python
def play_with_tensorflow_dataset_v1():

    # reset tensorflow
    tf.reset_default_graph()

    # create fake data
```

```
nsamples = 105

X = np.arange(nsamples, dtype='float32').reshape(nsamples, 1)
Y = np.ones((nsamples, 1), dtype='float32')

# loop / batch parameters
nepochs = 5
batchsize = 8
nbatches = int(np.ceil(nsamples/batchsize))

# Create a dataset from numpy array
dataset = tf.data.Dataset.from_tensor_slices((X, Y))
# Uncomment following line if you want to automatically shuffe the dataset
#dataset = dataset.shuffle(buffer_size=1000)
# Each iteration over the dataset will provide batchsize samples
dataset = dataset.batch(batchsize)

# Create an iterator over the dataset
dataset_iterator = dataset.make_initializable_iterator()

# Create placeholders for each dataset batch
xbatch, ybatch = dataset_iterator.get_next()

# Create an operation tree with these placeholders
f = tf.reduce_mean((xbatch-ybatch)**2)

init_global = tf.initializers.global_variables()

with tf.Session() as session:
    session.run(init_global)

    for epoch in range(nepochs):

        # Initialize the iterator
        session.run(dataset_iterator.initializer)
        # while True: would also work
        for b in range(nbatches):
            try:
                # get the next batch and run the f operation tree
                fval, xbatchval, ybatchval = session.run(
                    [f, xbatch, ybatch])
            except tf.errors.OutOfRangeError:
                break
            print("Epoch %d/%d Batch %d/%d " % (epoch+1, nepochs,
                                                b+1, nbatches), fval,␣
↪xbatchval.T, ybatchval.T)
```

```
play_with_tensorflow_dataset_v1()
```

```
Epoch 1/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 1/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 1/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 1/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 1/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 1/5 Batch 14/14  10609.0 [[104.]] [[1.]]
Epoch 2/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 2/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 2/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 2/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 2/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
```

```
Epoch 2/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 2/5 Batch 14/14  10609.0 [[104.]] [[1.]]
Epoch 3/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 3/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 3/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 3/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 3/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 3/5 Batch 14/14  10609.0 [[104.]] [[1.]]
Epoch 4/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 4/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 4/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 4/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 4/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
```

```
Epoch 4/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 4/5 Batch 14/14  10609.0 [[104.]] [[1.]]
Epoch 5/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 5/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 5/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 5/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 5/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 5/5 Batch 14/14  10609.0 [[104.]] [[1.]]
```

1.2 Multiple dataset

What should we do with multiple dataset ? For example how can we iterate over train and validation ?

[3]:
```
def play_with_tensorflow_dataset_v2():

    # reset tensorflow
    tf.reset_default_graph()

    # create fake data
    nsamples = 105

    Xtrain = np.arange(nsamples, dtype='float32').reshape(nsamples, 1)
    Ytrain = np.ones((nsamples, 1), dtype='float32')

    Xvalid = np.arange(nsamples, dtype='float32').reshape(nsamples, 1)[::-1]
    Yvalid = -np.ones((nsamples, 1), dtype='float32')

    # loop / batch parameters
    nepochs = 5
    batchsize = 8
    nbatches = int(np.ceil(nsamples/batchsize))

    # Create a tesnorflow dataset for each dataset
    traindataset = tf.data.Dataset.from_tensor_slices((Xtrain, Ytrain))
    traindataset = traindataset.batch(batchsize)

    validdataset = tf.data.Dataset.from_tensor_slices((Xvalid, Yvalid))
    validdataset = validdataset.batch(batchsize)

    # Create ONLY ONE iterator base on types and shapes of one of the dataset
    # both dataset should have the same types and shapes...
    dataset_iterator = tf.data.Iterator.from_structure(traindataset.
↪output_types,
                                                       traindataset.
↪output_shapes)

    # Create one initializer per dataset
    training_init_op = dataset_iterator.make_initializer(traindataset)
    validation_init_op = dataset_iterator .make_initializer(validdataset)

    # Create placeholders for each batch
    xbatch, ybatch = dataset_iterator.get_next()

    f = tf.reduce_mean((xbatch-ybatch)**2)

    init_global = tf.initializers.global_variables()
```

```
    with tf.Session() as session:
        session.run(init_global)

        for epoch in range(nepochs):
            print("== Train loop ==")
            # Initialize the training dataset iterator
            # Now xbatch,ybatch will iterate over the train dataset
            session.run(training_init_op)
            for b in range(nbatches):
                try:
                    fval, xbatchval, ybatchval = session.run(
                        [f, xbatch, ybatch])
                except tf.errors.OutOfRangeError:
                    break
                print("Epoch %d/%d Batch %d/%d " % (epoch+1, nepochs,
                                                    b+1, nbatches), fval,␣
 ↪xbatchval.T, ybatchval.T)
            print("== Validation loop ==")
            # Initialize the validation dataset iterator
            # Now xbatch,ybatch will iterate over the validation dataset
            session.run(validation_init_op)
            for b in range(nbatches):
                try:
                    fval, xbatchval, ybatchval = session.run(
                        [f, xbatch, ybatch])
                except tf.errors.OutOfRangeError:
                    break
                print("Epoch %d/%d Batch %d/%d " % (epoch+1, nepochs,
                                                    b+1, nbatches), fval,␣
 ↪xbatchval.T, ybatchval.T)


play_with_tensorflow_dataset_v2()
```

```
== Train loop ==
Epoch 1/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 1/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 1/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 1/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 1/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
```

```
1. 1. 1. 1.]]
Epoch 1/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 1/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 1/5 Batch 14/14  10609.0 [[104.]] [[1.]]
== Validation loop ==
Epoch 1/5 Batch 1/14  10307.5 [[104. 103. 102. 101. 100.  99.  98.  97.]] [[-1.
-1. -1. -1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 2/14  8747.5 [[96. 95. 94. 93. 92. 91. 90. 89.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 3/14  7315.5 [[88. 87. 86. 85. 84. 83. 82. 81.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 4/14  6011.5 [[80. 79. 78. 77. 76. 75. 74. 73.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 5/14  4835.5 [[72. 71. 70. 69. 68. 67. 66. 65.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 6/14  3787.5 [[64. 63. 62. 61. 60. 59. 58. 57.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 7/14  2867.5 [[56. 55. 54. 53. 52. 51. 50. 49.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 8/14  2075.5 [[48. 47. 46. 45. 44. 43. 42. 41.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 9/14  1411.5 [[40. 39. 38. 37. 36. 35. 34. 33.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 10/14  875.5 [[32. 31. 30. 29. 28. 27. 26. 25.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 11/14  467.5 [[24. 23. 22. 21. 20. 19. 18. 17.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 12/14  187.5 [[16. 15. 14. 13. 12. 11. 10.  9.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 1/5 Batch 13/14  35.5 [[8. 7. 6. 5. 4. 3. 2. 1.]] [[-1. -1. -1. -1. -1.
-1. -1. -1.]]
Epoch 1/5 Batch 14/14  1.0 [[0.]] [[-1.]]
== Train loop ==
Epoch 2/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 2/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
```

```
1. 1. 1.]]
Epoch 2/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 2/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 2/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 2/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 2/5 Batch 14/14  10609.0 [[104.]] [[1.]]
== Validation loop ==
Epoch 2/5 Batch 1/14  10307.5 [[104. 103. 102. 101. 100.  99.  98.  97.]] [[-1.
-1. -1. -1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 2/14  8747.5 [[96. 95. 94. 93. 92. 91. 90. 89.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 3/14  7315.5 [[88. 87. 86. 85. 84. 83. 82. 81.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 4/14  6011.5 [[80. 79. 78. 77. 76. 75. 74. 73.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 5/14  4835.5 [[72. 71. 70. 69. 68. 67. 66. 65.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 6/14  3787.5 [[64. 63. 62. 61. 60. 59. 58. 57.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 7/14  2867.5 [[56. 55. 54. 53. 52. 51. 50. 49.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 8/14  2075.5 [[48. 47. 46. 45. 44. 43. 42. 41.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 9/14  1411.5 [[40. 39. 38. 37. 36. 35. 34. 33.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 10/14  875.5 [[32. 31. 30. 29. 28. 27. 26. 25.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 11/14  467.5 [[24. 23. 22. 21. 20. 19. 18. 17.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 12/14  187.5 [[16. 15. 14. 13. 12. 11. 10.  9.]] [[-1. -1. -1.
```

```
-1. -1. -1. -1. -1.]]
Epoch 2/5 Batch 13/14  35.5 [[8. 7. 6. 5. 4. 3. 2. 1.]] [[-1. -1. -1. -1. -1.
-1. -1. -1.]]
Epoch 2/5 Batch 14/14  1.0 [[0.]] [[-1.]]
== Train loop ==
Epoch 3/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 3/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 3/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 3/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 3/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 3/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 3/5 Batch 14/14  10609.0 [[104.]] [[1.]]
== Validation loop ==
Epoch 3/5 Batch 1/14  10307.5 [[104. 103. 102. 101. 100.  99.  98.  97.]] [[-1.
-1. -1. -1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 2/14  8747.5 [[96. 95. 94. 93. 92. 91. 90. 89.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 3/14  7315.5 [[88. 87. 86. 85. 84. 83. 82. 81.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 4/14  6011.5 [[80. 79. 78. 77. 76. 75. 74. 73.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 5/14  4835.5 [[72. 71. 70. 69. 68. 67. 66. 65.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 6/14  3787.5 [[64. 63. 62. 61. 60. 59. 58. 57.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 7/14  2867.5 [[56. 55. 54. 53. 52. 51. 50. 49.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 8/14  2075.5 [[48. 47. 46. 45. 44. 43. 42. 41.]] [[-1. -1. -1.
```

```
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 9/14  1411.5 [[40. 39. 38. 37. 36. 35. 34. 33.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 10/14  875.5 [[32. 31. 30. 29. 28. 27. 26. 25.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 11/14  467.5 [[24. 23. 22. 21. 20. 19. 18. 17.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 12/14  187.5 [[16. 15. 14. 13. 12. 11. 10.  9.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 3/5 Batch 13/14  35.5 [[8. 7. 6. 5. 4. 3. 2. 1.]] [[-1. -1. -1. -1. -1.
-1. -1. -1.]]
Epoch 3/5 Batch 14/14  1.0 [[0.]] [[-1.]]
== Train loop ==
Epoch 4/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 4/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 4/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 4/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 4/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 4/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 4/5 Batch 14/14  10609.0 [[104.]] [[1.]]
== Validation loop ==
Epoch 4/5 Batch 1/14  10307.5 [[104. 103. 102. 101. 100.  99.  98.  97.]] [[-1.
-1. -1. -1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 2/14  8747.5 [[96. 95. 94. 93. 92. 91. 90. 89.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 3/14  7315.5 [[88. 87. 86. 85. 84. 83. 82. 81.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 4/14  6011.5 [[80. 79. 78. 77. 76. 75. 74. 73.]] [[-1. -1. -1.
```

```
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 5/14  4835.5 [[72. 71. 70. 69. 68. 67. 66. 65.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 6/14  3787.5 [[64. 63. 62. 61. 60. 59. 58. 57.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 7/14  2867.5 [[56. 55. 54. 53. 52. 51. 50. 49.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 8/14  2075.5 [[48. 47. 46. 45. 44. 43. 42. 41.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 9/14  1411.5 [[40. 39. 38. 37. 36. 35. 34. 33.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 10/14  875.5 [[32. 31. 30. 29. 28. 27. 26. 25.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 11/14  467.5 [[24. 23. 22. 21. 20. 19. 18. 17.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 12/14  187.5 [[16. 15. 14. 13. 12. 11. 10.  9.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 4/5 Batch 13/14  35.5 [[8. 7. 6. 5. 4. 3. 2. 1.]] [[-1. -1. -1. -1. -1.
-1. -1. -1.]]
Epoch 4/5 Batch 14/14  1.0 [[0.]] [[-1.]]
== Train loop ==
Epoch 5/5 Batch 1/14  11.5 [[0. 1. 2. 3. 4. 5. 6. 7.]] [[1. 1. 1. 1. 1. 1. 1.
1.]]
Epoch 5/5 Batch 2/14  115.5 [[ 8.  9. 10. 11. 12. 13. 14. 15.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 5/5 Batch 3/14  347.5 [[16. 17. 18. 19. 20. 21. 22. 23.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 5/5 Batch 4/14  707.5 [[24. 25. 26. 27. 28. 29. 30. 31.]] [[1. 1. 1. 1. 1.
1. 1. 1.]]
Epoch 5/5 Batch 5/14  1195.5 [[32. 33. 34. 35. 36. 37. 38. 39.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 6/14  1811.5 [[40. 41. 42. 43. 44. 45. 46. 47.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 7/14  2555.5 [[48. 49. 50. 51. 52. 53. 54. 55.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 8/14  3427.5 [[56. 57. 58. 59. 60. 61. 62. 63.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 9/14  4427.5 [[64. 65. 66. 67. 68. 69. 70. 71.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 10/14  5555.5 [[72. 73. 74. 75. 76. 77. 78. 79.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 11/14  6811.5 [[80. 81. 82. 83. 84. 85. 86. 87.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 12/14  8195.5 [[88. 89. 90. 91. 92. 93. 94. 95.]] [[1. 1. 1. 1.
1. 1. 1. 1.]]
Epoch 5/5 Batch 13/14  9707.5 [[ 96.  97.  98.  99. 100. 101. 102. 103.]] [[1.
1. 1. 1. 1. 1. 1. 1.]]
Epoch 5/5 Batch 14/14  10609.0 [[104.]] [[1.]]
```

```
== Validation loop ==
Epoch 5/5 Batch 1/14  10307.5 [[104. 103. 102. 101. 100.  99.  98.  97.]] [[-1.
-1. -1. -1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 2/14  8747.5 [[96. 95. 94. 93. 92. 91. 90. 89.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 3/14  7315.5 [[88. 87. 86. 85. 84. 83. 82. 81.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 4/14  6011.5 [[80. 79. 78. 77. 76. 75. 74. 73.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 5/14  4835.5 [[72. 71. 70. 69. 68. 67. 66. 65.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 6/14  3787.5 [[64. 63. 62. 61. 60. 59. 58. 57.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 7/14  2867.5 [[56. 55. 54. 53. 52. 51. 50. 49.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 8/14  2075.5 [[48. 47. 46. 45. 44. 43. 42. 41.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 9/14  1411.5 [[40. 39. 38. 37. 36. 35. 34. 33.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 10/14  875.5 [[32. 31. 30. 29. 28. 27. 26. 25.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 11/14  467.5 [[24. 23. 22. 21. 20. 19. 18. 17.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 12/14  187.5 [[16. 15. 14. 13. 12. 11. 10.  9.]] [[-1. -1. -1.
-1. -1. -1. -1. -1.]]
Epoch 5/5 Batch 13/14  35.5 [[8. 7. 6. 5. 4. 3. 2. 1.]] [[-1. -1. -1. -1. -1.
-1. -1. -1.]]
Epoch 5/5 Batch 14/14  1.0 [[0.]] [[-1.]]
```

1.3 Your turn !

Modify the initial linear regression function to introduce mini batches.

The typical code for computing the validation loss will be something like

```
def compute_loss(session,iterator_init,loss,nbatches):
    session.run(iterator_init)
    lossval = 0.
    for b in range(nbatches):
        batch_lossval, = session.run([loss])
        lossval += batch_lossval
    lossval /= nbatches
    return lossval
```

[4]:
```
def linear_regression_v2(trainset, validset, testset):

    Xtrain, Ytrain = trainset
    Xvalid, Yvalid = validset
    Xtest, Ytest = testset
```

```
# --- Linear Regression ---

# reset tensorflow
tf.reset_default_graph()

# Hyperparameters
learning_rate = 1e-2
training_epochs = 100
batchsize = 50

ntrain, nc = Xtrain.shape
ntest, _ = Xtest.shape
nvalid, _ = Xvalid.shape
_, no = Ytrain.shape

def prepareset(dataset, batchsize):
    dataset = dataset.shuffle(buffer_size=1000)
    dataset = dataset.batch(batchsize)
    return dataset

trainset = prepareset(tf.data.Dataset.from_tensor_slices(
    (Xtrain, Ytrain)), batchsize)
testset = prepareset(
    tf.data.Dataset.from_tensor_slices((Xtest, Ytest)), batchsize)
validset = prepareset(tf.data.Dataset.from_tensor_slices(
    (Xvalid, Yvalid)), batchsize)

ntrainbatches = int(np.ceil(ntrain/batchsize))
ntestbatches = int(np.ceil(ntest/batchsize))
nvalidbatches = int(np.ceil(nvalid/batchsize))

print((trainset.output_types,
       trainset.output_shapes))

# Create ONLY ONE iterator base on types and shapes of one of the dataset
# both dataset should have the same types and shapes...
dataset_iterator = tf.data.Iterator.from_structure(trainset.output_types,
                                                   trainset.output_shapes)

# Create one initializer per dataset
training_init_op = dataset_iterator.make_initializer(trainset)
test_init_op = dataset_iterator.make_initializer(testset)
validation_init_op = dataset_iterator .make_initializer(validset)

# Placeholders from the dataset iterator
x, y = dataset_iterator.get_next()
```

```
    # Linear regression parameters
    A = tf.Variable(tf.zeros([nc, no]))
    b = tf.Variable(tf.zeros([no]))
    # All parameters are gathered into var_list
    var_list = [A, b]

    # Actual linear regression
    pred = tf.matmul(x, A) + b

    # Model loss
    loss = tf.losses.mean_squared_error(y, pred)

    optimizer = tf.train.GradientDescentOptimizer(
        learning_rate).minimize(loss, var_list=var_list)

    init_global = tf.initializers.global_variables()

    def compute_loss(session, iterator_init, loss, nbatches):
        session.run(iterator_init)
        lossval = 0.
        for b in range(nbatches):
            batch_lossval, = session.run([loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    def compute_gradient_step(session, iterator_init, optimizer, loss,␣
↪nbatches):
        session.run(iterator_init)
        lossval = 0.
        for b in range(nbatches):
            _, batch_lossval, = session.run([optimizer, loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    with tf.Session() as session:

        # We call the initialization of A and b
        session.run(init_global)

        # We compute the train and validation loss
        # Note that you just have to change the feed_dict to change the set

        trainloss = compute_loss(
            session, training_init_op, loss, ntrainbatches)
```

```
        validationloss = compute_loss(
            session, validation_init_op, loss, nvalidbatches)
        print("Init\t\t train loss %f\t valid loss %f" %
              (trainloss, validationloss))

        # We cycle on epochs
        for epoch in range(training_epochs):

            trainloss = compute_gradient_step(
                session, training_init_op, optimizer, loss, ntrainbatches)
            validationloss = compute_loss(
                session, validation_init_op, loss, nvalidbatches)
            print("Epoch %03d\t train loss %f\t valid loss %f" %
                  (epoch, trainloss, validationloss))

        # We compute the test loss
        testloss = compute_loss(session, test_init_op, loss, ntestbatches)
        print("Test loss %f" % (testloss,))

        # Found parameters
        Aval, bval = session.run(var_list)
        print("Estimated A\n", Aval)
        print('Estimated b\n', bval)
    # Here session is closed automatically


linear_regression_v2(*generate_all_dataset())
```

```
True A
 [[ 1.76405235  0.40015721]
 [ 0.97873798  2.2408932 ]
 [ 1.86755799 -0.97727788]
 [ 0.95008842 -0.15135721]]
True b
 [-0.10321885  0.4105985 ]
((tf.float32, tf.float32), (TensorShape([Dimension(None), Dimension(4)]),
TensorShape([Dimension(None), Dimension(2)])))
Init             train loss 7.184081     valid loss 7.740232
Epoch 000        train loss 6.008697     valid loss 5.236967
Epoch 001        train loss 4.063022     valid loss 3.545145
Epoch 002        train loss 2.748372     valid loss 2.401219
Epoch 003        train loss 1.860185     valid loss 1.627297
Epoch 004        train loss 1.260208     valid loss 1.103390
Epoch 005        train loss 0.853830     valid loss 0.748583
Epoch 006        train loss 0.579057     valid loss 0.508128
Epoch 007        train loss 0.392849     valid loss 0.345097
Epoch 008        train loss 0.266700     valid loss 0.234503
```

```
Epoch 009       train loss 0.181130     valid loss 0.159438
Epoch 010       train loss 0.123137     valid loss 0.108454
Epoch 011       train loss 0.083735     valid loss 0.073812
Epoch 012       train loss 0.056989     valid loss 0.050257
Epoch 013       train loss 0.038777     valid loss 0.034241
Epoch 014       train loss 0.026422     valid loss 0.023338
Epoch 015       train loss 0.018004     valid loss 0.015916
Epoch 016       train loss 0.012280     valid loss 0.010859
Epoch 017       train loss 0.008378     valid loss 0.007412
Epoch 018       train loss 0.005718     valid loss 0.005062
Epoch 019       train loss 0.003906     valid loss 0.003459
Epoch 020       train loss 0.002669     valid loss 0.002364
Epoch 021       train loss 0.001825     valid loss 0.001617
Epoch 022       train loss 0.001248     valid loss 0.001106
Epoch 023       train loss 0.000854     valid loss 0.000757
Epoch 024       train loss 0.000585     valid loss 0.000519
Epoch 025       train loss 0.000401     valid loss 0.000355
Epoch 026       train loss 0.000275     valid loss 0.000244
Epoch 027       train loss 0.000188     valid loss 0.000167
Epoch 028       train loss 0.000129     valid loss 0.000115
Epoch 029       train loss 0.000089     valid loss 0.000079
Epoch 030       train loss 0.000061     valid loss 0.000054
Epoch 031       train loss 0.000042     valid loss 0.000037
Epoch 032       train loss 0.000029     valid loss 0.000025
Epoch 033       train loss 0.000020     valid loss 0.000017
Epoch 034       train loss 0.000014     valid loss 0.000012
Epoch 035       train loss 0.000009     valid loss 0.000008
Epoch 036       train loss 0.000006     valid loss 0.000006
Epoch 037       train loss 0.000004     valid loss 0.000004
Epoch 038       train loss 0.000003     valid loss 0.000003
Epoch 039       train loss 0.000002     valid loss 0.000002
Epoch 040       train loss 0.000001     valid loss 0.000001
Epoch 041       train loss 0.000001     valid loss 0.000001
Epoch 042       train loss 0.000001     valid loss 0.000001
Epoch 043       train loss 0.000000     valid loss 0.000000
Epoch 044       train loss 0.000000     valid loss 0.000000
Epoch 045       train loss 0.000000     valid loss 0.000000
Epoch 046       train loss 0.000000     valid loss 0.000000
Epoch 047       train loss 0.000000     valid loss 0.000000
Epoch 048       train loss 0.000000     valid loss 0.000000
Epoch 049       train loss 0.000000     valid loss 0.000000
Epoch 050       train loss 0.000000     valid loss 0.000000
Epoch 051       train loss 0.000000     valid loss 0.000000
Epoch 052       train loss 0.000000     valid loss 0.000000
Epoch 053       train loss 0.000000     valid loss 0.000000
Epoch 054       train loss 0.000000     valid loss 0.000000
Epoch 055       train loss 0.000000     valid loss 0.000000
Epoch 056       train loss 0.000000     valid loss 0.000000
```

```
Epoch 057        train loss 0.000000     valid loss 0.000000
Epoch 058        train loss 0.000000     valid loss 0.000000
Epoch 059        train loss 0.000000     valid loss 0.000000
Epoch 060        train loss 0.000000     valid loss 0.000000
Epoch 061        train loss 0.000000     valid loss 0.000000
Epoch 062        train loss 0.000000     valid loss 0.000000
Epoch 063        train loss 0.000000     valid loss 0.000000
Epoch 064        train loss 0.000000     valid loss 0.000000
Epoch 065        train loss 0.000000     valid loss 0.000000
Epoch 066        train loss 0.000000     valid loss 0.000000
Epoch 067        train loss 0.000000     valid loss 0.000000
Epoch 068        train loss 0.000000     valid loss 0.000000
Epoch 069        train loss 0.000000     valid loss 0.000000
Epoch 070        train loss 0.000000     valid loss 0.000000
Epoch 071        train loss 0.000000     valid loss 0.000000
Epoch 072        train loss 0.000000     valid loss 0.000000
Epoch 073        train loss 0.000000     valid loss 0.000000
Epoch 074        train loss 0.000000     valid loss 0.000000
Epoch 075        train loss 0.000000     valid loss 0.000000
Epoch 076        train loss 0.000000     valid loss 0.000000
Epoch 077        train loss 0.000000     valid loss 0.000000
Epoch 078        train loss 0.000000     valid loss 0.000000
Epoch 079        train loss 0.000000     valid loss 0.000000
Epoch 080        train loss 0.000000     valid loss 0.000000
Epoch 081        train loss 0.000000     valid loss 0.000000
Epoch 082        train loss 0.000000     valid loss 0.000000
Epoch 083        train loss 0.000000     valid loss 0.000000
Epoch 084        train loss 0.000000     valid loss 0.000000
Epoch 085        train loss 0.000000     valid loss 0.000000
Epoch 086        train loss 0.000000     valid loss 0.000000
Epoch 087        train loss 0.000000     valid loss 0.000000
Epoch 088        train loss 0.000000     valid loss 0.000000
Epoch 089        train loss 0.000000     valid loss 0.000000
Epoch 090        train loss 0.000000     valid loss 0.000000
Epoch 091        train loss 0.000000     valid loss 0.000000
Epoch 092        train loss 0.000000     valid loss 0.000000
Epoch 093        train loss 0.000000     valid loss 0.000000
Epoch 094        train loss 0.000000     valid loss 0.000000
Epoch 095        train loss 0.000000     valid loss 0.000000
Epoch 096        train loss 0.000000     valid loss 0.000000
Epoch 097        train loss 0.000000     valid loss 0.000000
Epoch 098        train loss 0.000000     valid loss 0.000000
Epoch 099        train loss 0.000000     valid loss 0.000000
Test loss 0.000000
Estimated A
 [[ 1.764049    0.4001572 ]
 [ 0.9787371   2.2408857 ]
 [ 1.8675544  -0.97727734]
```

```
 [ 0.95008725 -0.15135701]]
Estimated b
 [-0.10321921  0.41059837]
```

[5]:
```python
# with object oriented programming


class LinearRegressionV3:

    def __init__(self, dtype, ninputs, noutputs,
                 learning_rate=5e-3, training_epochs=100, batchsize=50):

        self.nc = ninputs
        self.no = noutputs

        self.dt_shapes = (tf.TensorShape((None, ninputs)),
                          tf.TensorShape((None, noutputs)))
        self.dt_types = (dtype, dtype)

        self.learning_rate = learning_rate
        self.training_epochs = training_epochs
        self.batchsize = batchsize

        self._build_network()

        # Global variable initializer
        self.init_global = tf.initializers.global_variables()

    def _build_network(self):
        # Create ONLY ONE iterator base on types and shapes of one of the␣
↪dataset
        # both dataset should have the same types and shapes...
        self.dataset_iterator = tf.data.Iterator.from_structure(
            self.dt_types, self.dt_shapes)

        # Placeholders from the dataset iterator
        x, y = self.dataset_iterator.get_next()

        # Linear regression parameters
        self.A = tf.Variable(tf.zeros([self.nc, self.no]))
        self.b = tf.Variable(tf.zeros([self.no]))
        # All parameters are gathered into var_list
        var_list = [self.A, self.b]

        # Actual linear regression
        self.pred = tf.matmul(x, self.A) + self.b
```

```
        # Model loss
        self.loss = tf.losses.mean_squared_error(y, self.pred)

        self.optimizer = tf.train.GradientDescentOptimizer(
            self.learning_rate).minimize(self.loss, var_list=var_list)

    def _prepareset(self, dataset, shuffle=True):
        if shuffle:
            dataset = dataset.shuffle(buffer_size=1000)
        dataset = dataset.batch(self.batchsize)
        return dataset

    def _compute_loss(self, set_iterator_init, nbatches):
        self.session.run(set_iterator_init)
        lossval = 0.
        for b in range(nbatches):
            batch_lossval, = self.session.run([self.loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    def _compute_gradient_step(self, set_iterator_init, nbatches):
        self.session.run(set_iterator_init)
        lossval = 0.
        for b in range(nbatches):
            _, batch_lossval, = self.session.run([self.optimizer, self.loss])
            lossval += batch_lossval
        lossval /= nbatches
        return lossval

    def train(self, trainset, validset, testset):

        (Xtrain, Ytrain) = trainset
        (Xvalid, Yvalid) = validset
        (Xtest, Ytest) = testset

        # --- Linear Regression ---

        ntrain, _ = Xtrain.shape
        ntest, _ = Xtest.shape
        nvalid, _ = Xvalid.shape

        trainset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtrain, Ytrain)))
        testset = self._prepareset(
            tf.data.Dataset.from_tensor_slices((Xtest, Ytest)), shuffle=False)
        validset = self._prepareset(
```

```
            tf.data.Dataset.from_tensor_slices((Xvalid, Yvalid)), shuffle=False)

        ntrainbatches = int(np.ceil(ntrain/self.batchsize))
        ntestbatches = int(np.ceil(ntest/self.batchsize))
        nvalidbatches = int(np.ceil(nvalid/self.batchsize))

        # Create one initializer per dataset
        training_init_op = self.dataset_iterator.make_initializer(trainset)
        test_init_op = self.dataset_iterator.make_initializer(testset)
        validation_init_op = self.dataset_iterator .make_initializer(validset)


        with tf.Session() as self.session:

            # We call the initialization of A and b
            self.session.run(self.init_global)

            # We compute the train and validation loss
            # Note that you just have to change the feed_dict to change the set

            trainloss = self._compute_loss(training_init_op, ntrainbatches)
            validationloss = self._compute_loss(
                validation_init_op, nvalidbatches)
            print("Init\t\t train loss %f\t valid loss %f" %
                  (trainloss, validationloss))

            # We cycle on epochs
            for epoch in range(self.training_epochs):

                trainloss = self._compute_gradient_step(
                    training_init_op, ntrainbatches)
                validationloss = self._compute_loss(
                    validation_init_op,  nvalidbatches)
                print("Epoch %03d\t train loss %f\t valid loss %f" %
                      (epoch+1, trainloss, validationloss))

            # We compute the test loss
            testloss = self._compute_loss(test_init_op, ntestbatches)
            print("Test loss %f" % (testloss,))

            # Found parameters
            Aval, bval = self.session.run([self.A, self.b])
            print("Estimated A\n", Aval)
            print('Estimated b\n', bval)
        # Here session is closed automatically
```

```python
def linear_regression_v3(trainset, validset, testset):
    # reset tensorflow
    tf.reset_default_graph()

    Xtrain, Ytrain = trainset
    _, ninputs = Xtrain.shape
    _, noutputs = Ytrain.shape

    lr = LinearRegressionV3(tf.float32, ninputs, noutputs)
    lr.train(trainset, validset, testset)


linear_regression_v3(*generate_all_dataset())
```

```
True A
 [[ 1.76405235  0.40015721]
 [ 0.97873798  2.2408932 ]
 [ 1.86755799 -0.97727788]
 [ 0.95008842 -0.15135721]]
True b
 [-0.10321885  0.4105985 ]
Init             train loss 7.184081     valid loss 7.740232
Epoch 001        train loss 6.561208     valid loss 6.369670
Epoch 002        train loss 5.397117     valid loss 5.242507
Epoch 003        train loss 4.440261     valid loss 4.315373
Epoch 004        train loss 3.653582     valid loss 3.552691
Epoch 005        train loss 3.006378     valid loss 2.925221
Epoch 006        train loss 2.474727     valid loss 2.408885
Epoch 007        train loss 2.037167     valid loss 1.983950
Epoch 008        train loss 1.677013     valid loss 1.634213
Epoch 009        train loss 1.380905     valid loss 1.346309
Epoch 010        train loss 1.137252     valid loss 1.109274
Epoch 011        train loss 0.936770     valid loss 0.914091
Epoch 012        train loss 0.771717     valid loss 0.753347
Epoch 013        train loss 0.635810     valid loss 0.620953
Epoch 014        train loss 0.523950     valid loss 0.511892
Epoch 015        train loss 0.431766     valid loss 0.422045
Epoch 016        train loss 0.355889     valid loss 0.348012
Epoch 017        train loss 0.293369     valid loss 0.287006
Epoch 018        train loss 0.241877     valid loss 0.236725
Epoch 019        train loss 0.199478     valid loss 0.195276
Epoch 020        train loss 0.164488     valid loss 0.161108
Epoch 021        train loss 0.135694     valid loss 0.132934
Epoch 022        train loss 0.111964     valid loss 0.109699
Epoch 023        train loss 0.092347     valid loss 0.090539
Epoch 024        train loss 0.076212     valid loss 0.074735
Epoch 025        train loss 0.062895     valid loss 0.061697
```

```
Epoch 026       train loss 0.051914     valid loss 0.050940
Epoch 027       train loss 0.042859     valid loss 0.042064
Epoch 028       train loss 0.035387     valid loss 0.034739
Epoch 029       train loss 0.029221     valid loss 0.028692
Epoch 030       train loss 0.024130     valid loss 0.023702
Epoch 031       train loss 0.019932     valid loss 0.019582
Epoch 032       train loss 0.016465     valid loss 0.016180
Epoch 033       train loss 0.013603     valid loss 0.013370
Epoch 034       train loss 0.011240     valid loss 0.011050
Epoch 035       train loss 0.009289     valid loss 0.009134
Epoch 036       train loss 0.007679     valid loss 0.007550
Epoch 037       train loss 0.006347     valid loss 0.006242
Epoch 038       train loss 0.005247     valid loss 0.005161
Epoch 039       train loss 0.004338     valid loss 0.004268
Epoch 040       train loss 0.003587     valid loss 0.003530
Epoch 041       train loss 0.002967     valid loss 0.002920
Epoch 042       train loss 0.002454     valid loss 0.002415
Epoch 043       train loss 0.002031     valid loss 0.001998
Epoch 044       train loss 0.001680     valid loss 0.001653
Epoch 045       train loss 0.001390     valid loss 0.001368
Epoch 046       train loss 0.001150     valid loss 0.001132
Epoch 047       train loss 0.000952     valid loss 0.000937
Epoch 048       train loss 0.000788     valid loss 0.000776
Epoch 049       train loss 0.000652     valid loss 0.000642
Epoch 050       train loss 0.000540     valid loss 0.000532
Epoch 051       train loss 0.000447     valid loss 0.000440
Epoch 052       train loss 0.000370     valid loss 0.000365
Epoch 053       train loss 0.000307     valid loss 0.000302
Epoch 054       train loss 0.000254     valid loss 0.000250
Epoch 055       train loss 0.000210     valid loss 0.000207
Epoch 056       train loss 0.000174     valid loss 0.000172
Epoch 057       train loss 0.000145     valid loss 0.000142
Epoch 058       train loss 0.000120     valid loss 0.000118
Epoch 059       train loss 0.000099     valid loss 0.000098
Epoch 060       train loss 0.000082     valid loss 0.000081
Epoch 061       train loss 0.000068     valid loss 0.000067
Epoch 062       train loss 0.000057     valid loss 0.000056
Epoch 063       train loss 0.000047     valid loss 0.000046
Epoch 064       train loss 0.000039     valid loss 0.000038
Epoch 065       train loss 0.000032     valid loss 0.000032
Epoch 066       train loss 0.000027     valid loss 0.000026
Epoch 067       train loss 0.000022     valid loss 0.000022
Epoch 068       train loss 0.000018     valid loss 0.000018
Epoch 069       train loss 0.000015     valid loss 0.000015
Epoch 070       train loss 0.000013     valid loss 0.000012
Epoch 071       train loss 0.000011     valid loss 0.000010
Epoch 072       train loss 0.000009     valid loss 0.000009
Epoch 073       train loss 0.000007     valid loss 0.000007
```

```
Epoch 074        train loss 0.000006     valid loss 0.000006
Epoch 075        train loss 0.000005     valid loss 0.000005
Epoch 076        train loss 0.000004     valid loss 0.000004
Epoch 077        train loss 0.000003     valid loss 0.000003
Epoch 078        train loss 0.000003     valid loss 0.000003
Epoch 079        train loss 0.000002     valid loss 0.000002
Epoch 080        train loss 0.000002     valid loss 0.000002
Epoch 081        train loss 0.000002     valid loss 0.000002
Epoch 082        train loss 0.000001     valid loss 0.000001
Epoch 083        train loss 0.000001     valid loss 0.000001
Epoch 084        train loss 0.000001     valid loss 0.000001
Epoch 085        train loss 0.000001     valid loss 0.000001
Epoch 086        train loss 0.000001     valid loss 0.000001
Epoch 087        train loss 0.000001     valid loss 0.000001
Epoch 088        train loss 0.000000     valid loss 0.000000
Epoch 089        train loss 0.000000     valid loss 0.000000
Epoch 090        train loss 0.000000     valid loss 0.000000
Epoch 091        train loss 0.000000     valid loss 0.000000
Epoch 092        train loss 0.000000     valid loss 0.000000
Epoch 093        train loss 0.000000     valid loss 0.000000
Epoch 094        train loss 0.000000     valid loss 0.000000
Epoch 095        train loss 0.000000     valid loss 0.000000
Epoch 096        train loss 0.000000     valid loss 0.000000
Epoch 097        train loss 0.000000     valid loss 0.000000
Epoch 098        train loss 0.000000     valid loss 0.000000
Epoch 099        train loss 0.000000     valid loss 0.000000
Epoch 100        train loss 0.000000     valid loss 0.000000
Test loss 0.000000
Estimated A
 [[ 1.7639108   0.40013558]
 [ 0.9786886   2.2407737 ]
 [ 1.8673928  -0.9772391 ]
 [ 0.9499644  -0.15130529]]
Estimated b
 [-0.10332616  0.41058993]
```