DLM_basic

January 29, 2019

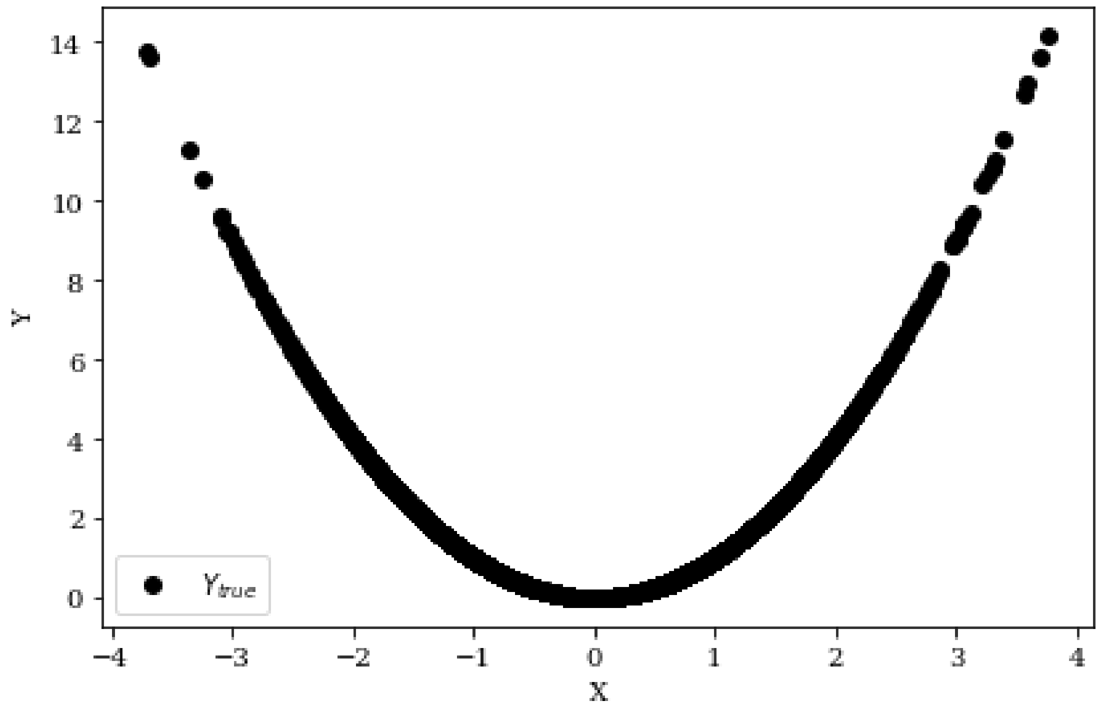
In [1]:
```
# IMPORT PACKAGES

from keras.layers import Input, Dense, Dropout
from keras.models import Model
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, CSVLogger
from sklearn.metrics import mean_squared_error
import numpy as np
from matplotlib import rcParams  # next 3 lines set font family for plotting
rcParams['font.family'] = 'serif'
rcParams['font.sans-serif'] = ['TImes New Roman']
import matplotlib.pyplot as plt
import os
import time
```

In [2]:
```
# SETTINGS FOR REPRODUCIBLE RESULTS DURING DEVELOPMENT

#import numpy as np
import tensorflow as tf
import random as rn

# The below is necessary in Python 3.2.3 onwards to
# have reproducible behavior for certain hash-based operations.
# See these references for further details:
# https://docs.python.org/3.4/using/cmdline.html#envvar-PYTHONHASHSEED
# https://github.com/keras-team/keras/issues/2280#issuecomment-306959926

#import os
os.environ['PYTHONHASHSEED'] = '0'

# The below is necessary for starting Numpy generated random numbers
# in a well-defined initial state.

np.random.seed(42)

# The below is necessary for starting core Python generated random numbers
# in a well-defined state.

rn.seed(12345)
```

```
# Force TensorFlow to use single thread.
# Multiple threads are a potential source of
# non-reproducible results.
# For further details, see: https://stackoverflow.com/questions/42022950/which-seeds-hav

session_conf = tf.ConfigProto(intra_op_parallelism_threads=1, inter_op_parallelism_threa

from keras import backend as K

# The below tf.set_random_seed() will make random number generation
# in the TensorFlow backend have a well-defined initial state.
# For further details, see: https://www.tensorflow.org/api_docs/python/tf/set_random_see

tf.set_random_seed(1234)

sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

In [3]:
```
# Create some simple data for a known function (y = x^2)

# generate 100,000 samples (x dimension) with 1 features each (y dimension)
X = np.random.randn(100000,1)
Y = np.square(X)
X_Nfeatures = X.shape[1]
Y_Nfeatures = Y.shape[1]

# partition X and Y into training (90%) and test (10%) sets
X_train = X[:90000,:]
X_test = X[90000:,:]
Y_train = Y[:90000,:]
Y_test = Y[90000:,:]

print('X dimensions:',X.shape)
print('Y dimensions:',Y.shape)

fig = plt.figure(num=1)
ax = fig.add_subplot(111)
y_true = ax.plot(X_test,Y_test,'ko',label=r'$Y_{true}$')
ax.set_xlabel('X')
ax.set_ylabel('Y')
curves = y_true
labels = [c.get_label() for c in curves]
ax.legend(curves, labels, loc=0)
plt.tight_layout()
plt.savefig('ytrue.pdf')
plt.show()
```

```
X dimensions: (100000, 1)
Y dimensions: (100000, 1)
```



In [4]:
```
# Build the DLM

tic = time.time()  # start a timer

# HYPERPARAMETERS
epochs = 5000
batch_size = 5000
do = 0.2
N_nodes = 64

# create input layer..........
main_input = Input(shape=(X_Nfeatures),
                   dtype='float',
                   batch_shape=(batch_size,X_Nfeatures),
                   name='main_input'
                   )
#create hidden layer..........
hidden_layer1 = Dense(N_nodes, activation='relu', name='hidden_layer1')(main_input)
# add dropout to hidden layer
Dropout(do)(hidden_layer1)
```

```
hidden_layer2 = Dense(N_nodes, activation='relu', name='hidden_layer2')(hidden_layer1)
# add dropout to hidden layer
Dropout(do)(hidden_layer2)

# create output layer
main_output = Dense(Y_Nfeatures, name='main_output')(hidden_layer2)  # default
activation is linear

# feed datasets into model for training
model = Model(inputs=[main_input],
              outputs=[main_output]
              )

# compile the model with desired configuration
model.compile(loss='mean_squared_error',
              optimizer='adagrad',
              metrics=['mae'])

# one of several callbacks available in Keras, csv_logger saves metrics for every epoch
csv_logger = CSVLogger('training_' + str(epochs) + '.log')

early_stop = EarlyStopping(monitor='val_loss', # quantity to monitor
                           min_delta=0.0001,  # min change to qualify as an improvement
                           patience=10, # stop after #epochs with no improvement
                           verbose=1)  # print messages

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,  # reduction factor (new_lr = lr * factor)
                              patience=5,
                              verbose=1)

# train the model, and store training information in the history object
history = model.fit([X_train],[Y_train],
                    epochs=epochs,
                    batch_size = batch_size,
                    validation_data=(X_test, Y_test),
                    callbacks=[reduce_lr,early_stop,csv_logger]
                    )
model.summary()  # print out a summary of layers/parameters
config = model.get_config()  # detailed information about the configuration
of each layer

# evaluate the trained model on the test data set
test = model.evaluate([X_test],[Y_test],batch_size=batch_size)
names = model.metrics_names

X_pred = np.random.randn(100000,1)
```

```python
        Y_pred = np.square(X_pred)
        predict = model.predict([X_pred],batch_size=batch_size)
        Y_mse = mean_squared_error(predict,Y_pred)
        print('Y_mse:',Y_mse)
```

```
Train on 90000 samples, validate on 10000 samples
Epoch 1/5000
90000/90000 [==============================] - 0s 5us/step - loss: 1.1451 - mean_absolute_error:
Epoch 2/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.4322 - mean_absolute_error:
Epoch 3/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.2532 - mean_absolute_error:
Epoch 4/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.1642 - mean_absolute_error:
Epoch 5/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.1159 - mean_absolute_error:
Epoch 6/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0866 - mean_absolute_error:
Epoch 7/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0676 - mean_absolute_error:
Epoch 8/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0547 - mean_absolute_error:
Epoch 9/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0453 - mean_absolute_error:
Epoch 10/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0384 - mean_absolute_error:
Epoch 11/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0330 - mean_absolute_error:
Epoch 12/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0287 - mean_absolute_error:
Epoch 13/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0252 - mean_absolute_error:
Epoch 14/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0224 - mean_absolute_error:
Epoch 15/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0201 - mean_absolute_error:
Epoch 16/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0181 - mean_absolute_error:
Epoch 17/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0165 - mean_absolute_error:
Epoch 18/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0151 - mean_absolute_error:
Epoch 19/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0139 - mean_absolute_error:
Epoch 20/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0129 - mean_absolute_error:
Epoch 21/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0119 - mean_absolute_error:
```

```
Epoch 22/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0111 - mean_absolute_error:
Epoch 23/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0105 - mean_absolute_error:
Epoch 24/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0098 - mean_absolute_error:
Epoch 25/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0092 - mean_absolute_error:
Epoch 26/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0087 - mean_absolute_error:
Epoch 27/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0081 - mean_absolute_error:
Epoch 28/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0078 - mean_absolute_error:
Epoch 29/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0074 - mean_absolute_error:
Epoch 30/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0070 - mean_absolute_error:
Epoch 31/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0067 - mean_absolute_error:
Epoch 32/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0064 - mean_absolute_error:
Epoch 33/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0061 - mean_absolute_error:
Epoch 34/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0058 - mean_absolute_error:
Epoch 35/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0056 - mean_absolute_error:
Epoch 36/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0054 - mean_absolute_error:
Epoch 37/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0052 - mean_absolute_error:
Epoch 38/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0050 - mean_absolute_error:
Epoch 39/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0048 - mean_absolute_error:
Epoch 40/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0046 - mean_absolute_error:
Epoch 41/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0045 - mean_absolute_error:
Epoch 42/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0043 - mean_absolute_error:
Epoch 43/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0043 - mean_absolute_error:
Epoch 44/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0041 - mean_absolute_error:
Epoch 45/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0040 - mean_absolute_error:
```

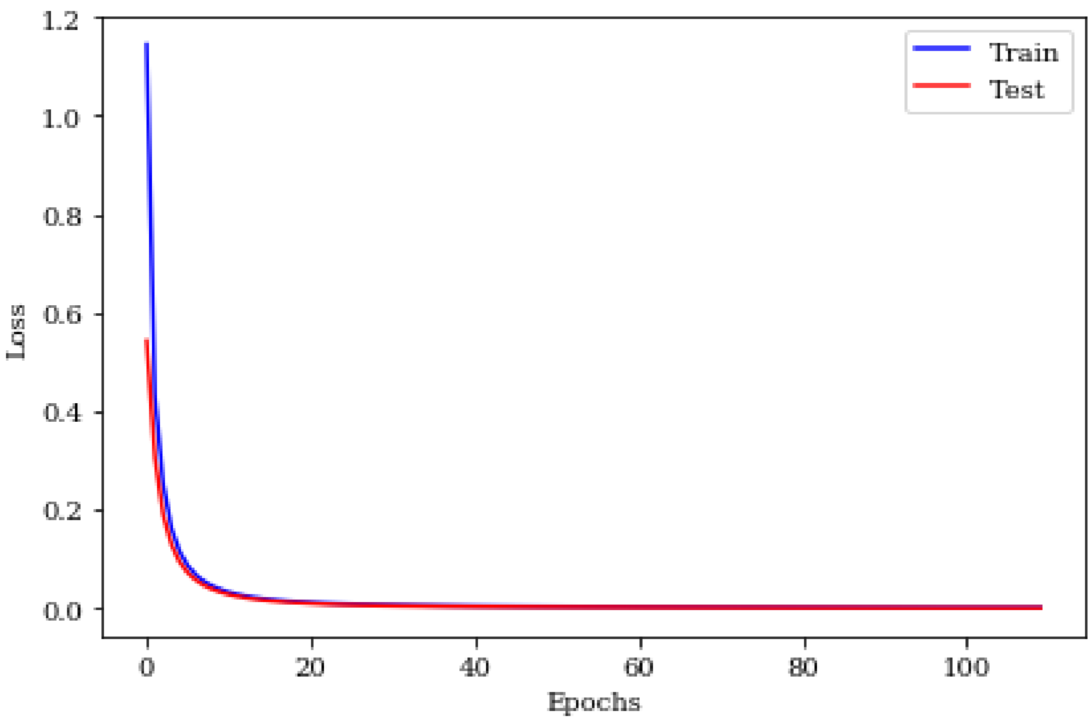
```
Epoch 46/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0038 - mean_absolute_error:
Epoch 47/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0037 - mean_absolute_error:
Epoch 48/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0036 - mean_absolute_error:
Epoch 49/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0035 - mean_absolute_error:
Epoch 50/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0034 - mean_absolute_error:
Epoch 51/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0033 - mean_absolute_error:
Epoch 52/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0032 - mean_absolute_error:
Epoch 53/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0032 - mean_absolute_error:
Epoch 54/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0031 - mean_absolute_error:
Epoch 55/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0030 - mean_absolute_error:
Epoch 56/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0029 - mean_absolute_error:
Epoch 57/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0029 - mean_absolute_error:
Epoch 58/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0028 - mean_absolute_error:
Epoch 59/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0028 - mean_absolute_error:
Epoch 60/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0027 - mean_absolute_error:
Epoch 61/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0026 - mean_absolute_error:
Epoch 62/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0026 - mean_absolute_error:
Epoch 63/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0025 - mean_absolute_error:
Epoch 64/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0025 - mean_absolute_error:
Epoch 65/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0024 - mean_absolute_error:
Epoch 66/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0024 - mean_absolute_error:
Epoch 67/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0023 - mean_absolute_error:
Epoch 68/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0023 - mean_absolute_error:
Epoch 69/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0022 - mean_absolute_error:
```

```
Epoch 70/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0022 - mean_absolute_error:
Epoch 71/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0021 - mean_absolute_error:
Epoch 72/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0021 - mean_absolute_error:
Epoch 73/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0021 - mean_absolute_error:
Epoch 74/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0020 - mean_absolute_error:
Epoch 75/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0020 - mean_absolute_error:
Epoch 76/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0020 - mean_absolute_error:
Epoch 77/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0019 - mean_absolute_error:
Epoch 78/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0019 - mean_absolute_error:
Epoch 79/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0019 - mean_absolute_error:
Epoch 80/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0018 - mean_absolute_error:
Epoch 81/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0018 - mean_absolute_error:
Epoch 82/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0018 - mean_absolute_error:
Epoch 83/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0017 - mean_absolute_error:
Epoch 84/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0017 - mean_absolute_error:
Epoch 85/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0017 - mean_absolute_error:
Epoch 86/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0017 - mean_absolute_error:
Epoch 87/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0017 - mean_absolute_error:
Epoch 88/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0016 - mean_absolute_error:
Epoch 89/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0016 - mean_absolute_error:
Epoch 90/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0016 - mean_absolute_error:
Epoch 91/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0016 - mean_absolute_error:
Epoch 92/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0015 - mean_absolute_error:
Epoch 93/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0015 - mean_absolute_error:
```

```
Epoch 94/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0015 - mean_absolute_error:
Epoch 95/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0015 - mean_absolute_error:
Epoch 96/5000
70000/90000 [======================>...] - ETA: 0s - loss: 0.0014 - mean_absolute_error: 0.0066
Epoch 00096: reducing learning rate to 0.0019999999552965165.
90000/90000 [==============================] - 0s 1us/step - loss: 0.0015 - mean_absolute_error:
Epoch 97/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 98/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 99/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 100/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 101/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 102/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 103/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 104/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 105/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 106/5000
75000/90000 [========================>...] - ETA: 0s - loss: 0.0012 - mean_absolute_error: 0.005
Epoch 00106: reducing learning rate to 0.0003999999724328518.
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 107/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 108/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 109/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 110/5000
90000/90000 [==============================] - 0s 1us/step - loss: 0.0014 - mean_absolute_error:
Epoch 00110: early stopping
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
main_input (InputLayer)      (5000, 1)                 0
_________________________________________________________________
hidden_layer1 (Dense)        (5000, 64)                128
_________________________________________________________________
hidden_layer2 (Dense)        (5000, 64)                4160
_________________________________________________________________
main_output (Dense)          (5000, 1)                 65
=================================================================
Total params: 4,353
Trainable params: 4,353
Non-trainable params: 0
_________________________________________________________________
10000/10000 [==============================] - 0s 1us/step
Y_mse: 0.001008258246466293
```

In [5]:
```
loss_train = history.history['loss']
loss_test = history.history['val_loss']
xplot = list(range(len(loss_train)))

fig = plt.figure(num=2)
ax = fig.add_subplot(111)
train = ax.plot(xplot,loss_train,'b-',label='Train')
test = ax.plot(xplot,loss_test,'r-',label='Test')
ax.set_xlabel('Epochs')
ax.set_ylabel('Loss')
curves = train+test
labels = [c.get_label() for c in curves]
ax.legend(curves, labels, loc=0)
plt.tight_layout()
plt.savefig('loss' + str(epochs) + 'epochs.pdf')
plt.show()
```



```
fig = plt.figure(num=3)
ax = fig.add_subplot(111)
y_true = ax.plot(X_pred,Y_pred,'ko',label=r'$Y_{true}$')
y_pred = ax.plot(X_pred,predict,'m*',label=r'$\hat{Y}$')
ax.set_xlabel(r'$X$')
ax.set_ylabel(r'$Y$')
curves = y_true+y_pred
labels = [c.get_label() for c in curves]
ax.legend(curves, labels, loc=0)
plt.tight_layout()
plt.savefig('ypred.pdf')
plt.show()
```