Robotics II: Control, Modeling and Learning

Laboratory 6

By Zarema Balgabekova

In [278…
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
from keras.utils import np_utils
from keras import backend as K
```

The dataset of 5000 points was collected using dataset.py.

In [279…
```python
#dict.csv contains 5000 data points
data = pd.read_csv("dict2.csv", header = None, names = ["Angles", "XY"])
```

In [280…
```python
train = data['Angles'].to_numpy()
labels = data['XY'].to_numpy()
```

In [282…
```python
X = list()
Y = list()
for i in range(len(train)):
    labels[i] = labels[i].replace('   ', ' ')
    labels[i] = labels[i].replace('  ', ' ')
    labels[i] = labels[i].strip('[ ').strip(' ]')
    train[i] = train[i].strip('(').strip(')')
    result = [float(val) for val in train[i].split(',')]
    X.append(result)
    result = [float(val) for val in labels[i].split(' ')]
    Y.append(result)
```

In [283…
```python
def rmse(y_true, y_pred):
    return K.sqrt(K.mean(K.square(y_pred - y_true)))
```

Trying different regression losses

1 RMSE (Root mean squared error)

In [284…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss=rmse, optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("RMSE: %.6f" % (scores))
```

```
RMSE: 0.062683
```

2 MSE (Mean squared error)

In [285…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSE: %.6f" % (scores))
```

```
MSE: 0.004353
```

3 MAE (Mean absolute error)

In [286…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_absolute_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MAE: %.6f" % (scores))
```

```
MAE: 0.052243
```

4 MSLE (Mean squared logarithmic error)

In [287…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000465
```

5 Huber loss

In [288…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='huber', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("Huber: %.6f" % (scores))
```

```
Huber: 0.001976
```

6 Log cosh

In [289…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='log_cosh', optimizer=Adam(0.01))


X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("Log cosh: %.6f" % (scores))
```

```
Log cosh: 0.001451
```

Therefore, MSLE loss should be used.

Trying different optimizers

1 Adam

For Adam optimizer, MSLE = 0.000465.

In [290…
```python
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import Adadelta
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import Adagrad
from tensorflow.keras.optimizers import Adamax
from tensorflow.keras.optimizers import Nadam
from tensorflow.keras.optimizers import Ftrl
```

2 SGD

In [291…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=SGD(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.050566
```

3 Adadelta

In [292…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adadelta(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.744011
```

4 RMSprop

In [293…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=RMSprop(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.001378
```

5 Adagrad

In [294…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adagrad(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.006846
```

6 Adamax

In [295…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adamax(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000792
```

7 Nadam

In [296…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Nadam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000735
```

8 Ftrl

In [297…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Ftrl(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.082130
```

Therefore, Adam optimizer should be used. However, Nadam and Adamax also perform well.

Changing layers

1 Changing the number of neurons for the first hidden layer

In [298…
```python
model = Sequential()
model.add(Dense(32, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))


X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000516
```

In [299…
```python
model = Sequential()
model.add(Dense(64, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))


X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000842
```

In [303…
```python
model = Sequential()
model.add(Dense(5, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))


X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.001028
```

We can see that the result does not improve. That's why we will keep 10 neurons for the first layer.

2 Adding more hiddenlayers

In [308…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(8, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000399
```

In [307…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(4, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000543
```

In [310…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(8, activation = 'relu'))
model.add(Dense(4, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000472
```

It is seen that MSLE is the smallest when we add one more hidden layer with 8 neurons.

3 Decreasing the number of hidden layers

In [315…
```python
model = Sequential()
model.add(Dense(10, input_dim =5, activation = 'relu'))
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.000766
```

In [316…
```python
model = Sequential()
model.add(Dense(2, activation='linear'))
model.compile(loss='mean_squared_logarithmic_error', optimizer=Adam(0.01))

X_train, X_test, y_train, y_test = train_test_split(np.asarray(X), np.asarray(Y), test_size=0.20)
y_train = np.delete(y_train, 2, 1)
y_test = np.delete(y_test, 2, 1)

model.fit(X_train, y_train, epochs = 15, verbose = 0)
scores = model.evaluate(X_test, y_test, verbose = 0)
print("MSLE: %.6f" % (scores))
```

```
MSLE: 0.005244
```

Decreasing the number of layers does not improve the result.

Results

By using the provided parameters (neural network with two hidden layers, RMSE loss, and Adam optimizer), we obtained the following loss: $\mathbf{RMSE = 0.062683}$. It was discovered that MSLE significantly decreases the loss. Trying different optimizers, we came to conclusion that Adam optimizer gives the best performance. Furthermore, the addition of one more hidden layer also improved the result. The best loss obtained is $\mathbf{MSLE = 0.000399}$. Therefore, the result was improved by 157 times.