Using models

Puteaux, Fall/Winter 2020-2021

```
################################
##                            ##
##  Deep Learning in Python   ##
##                            ##
################################
```

```
§1 Introduction to Deep Learning in Python
```

```
§1.3 Building deep learning models with keras
```

```
§1.3.4 Using models
```

1. How to use models?

- Save.
- Reload.
- Make predictions.

2. Code of saving, reloading, and using the model reloaded:

[1]:
```
import pandas as pd
from keras.layers import Dense
from keras.models import Sequential
from keras.utils.np_utils import to_categorical

data = pd.read_csv('ref5. Basketball shot log.csv')


def Data_preparation(df):
    df = df.reindex(columns=[
        'SHOT_CLOCK', 'DRIBBLES', 'TOUCH_TIME', 'SHOT_DIST', 'CLOSE_DEF_DIST',
        'SHOT_RESULT'
    ])
    df['SHOT_CLOCK'] = df['SHOT_CLOCK'].fillna(0)
    df['SHOT_RESULT'].replace('missed', 0, inplace=True)
    df['SHOT_RESULT'].replace('made', 1, inplace=True)
    df.columns = df.columns.str.lower()
    return df
```

```python
df = Data_preparation(data)
predictors = df.drop(['shot_result'], axis=1).to_numpy()
n_cols = predictors.shape[1]
target = to_categorical(df.shot_result)

model = Sequential()
model.add(Dense(100, activation='relu', input_shape=(n_cols, )))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(predictors, target)
```

```
4003/4003 [==============================] - 6s 2ms/step - loss: 0.6673 -
accuracy: 0.6037
```

[1]:
```
<tensorflow.python.keras.callbacks.History at 0x7fd6b839b990>
```

[2]:
```python
from keras.models import load_model

model.save('ref7. Model file.h5')
my_model = load_model('ref7. Model file.h5')

predictions = my_model.predict(predictors)
probability_true = predictions[:, 1]
probability_true
```

[2]:
```
array([0.4239784 , 0.28861964, 0.34507743, …, 0.3723459 , 0.34746826,
       0.4257349 ], dtype=float32)
```

[3]:
```python
my_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               600
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 202
=================================================================
```

```
Total params: 21,002
Trainable params: 21,002
Non-trainable params: 0
_________________________________________________________________
```

3. Practice exercises for using models:

▶ **Package pre-loading:**

[4]:
```python
import pandas as pd
import numpy as np
import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
```

▶ **Data pre-loading:**

[5]:
```python
df = pd.read_csv('ref6. Titanic.csv')

df.replace(False, 0, inplace=True)
df.replace(True, 1, inplace=True)

predictors = df.drop(['survived'], axis=1).to_numpy()
n_cols = predictors.shape[1]
target = to_categorical(df.survived)

pred_data = pd.read_csv('ref8. Titanic predictors data.csv')
pred_data.replace(False, 0, inplace=True)
pred_data.replace(True, 1, inplace=True)
```

▶ **Making predictions practice:**

[6]:
```python
# Specify, compile, and fit the model
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(n_cols, )))
model.add(Dense(2, activation='softmax'))
model.compile(optimizer='sgd',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(predictors, target)

# Calculate predictions: predictions
predictions = model.predict(pred_data)

# Calculate predicted probability of survival: predicted_prob_true
predicted_prob_true = predictions[:, 1]

# print predicted_prob_true
```

```python
print(predicted_prob_true)
```

```
28/28 [==============================] - 1s 19ms/step - loss: 6.0198 - accuracy:
0.5309
[2.26931944e-02 7.22933471e-01 1.00000000e+00 9.32792664e-01
 1.48448013e-02 8.83448124e-03 5.09101199e-03 1.00313820e-01
 8.63493141e-03 9.99853969e-01 2.02060249e-02 1.61479101e-01
 9.86266695e-03 9.98783052e-01 9.96851455e-03 9.70405806e-03
 4.08334918e-02 9.89836752e-01 1.45879586e-03 9.95671272e-01
 9.99999881e-01 1.96965951e-02 5.71862515e-03 6.31486103e-02
 9.99855757e-01 8.44954886e-03 9.99950051e-01 9.99822557e-01
 9.62737203e-03 9.99972820e-01 7.66332567e-01 9.85070705e-01
 8.00327025e-03 3.53872925e-02 1.12161405e-01 9.99999881e-01
 6.83216900e-02 8.03738739e-03 9.99934196e-01 9.07610655e-01
 6.70859367e-02 3.35104197e-01 9.56307590e-01 5.35367383e-03
 1.50574729e-01 1.83323678e-03 9.99998093e-01 5.75761683e-03
 9.70815301e-01 1.00000000e+00 9.98729289e-01 3.98600387e-05
 9.49907005e-01 9.99448597e-01 1.58789724e-01 1.50712267e-01
 1.00000000e+00 4.90621105e-02 5.02465606e-01 8.00327025e-03
 6.95733540e-03 2.19806105e-01 7.90199116e-02 9.99998450e-01
 1.66200370e-01 5.52499248e-03 1.49134636e-01 9.99702156e-01
 1.09818364e-02 9.00185108e-01 2.02369187e-02 9.96572971e-01
 1.05893016e-02 1.07396080e-03 7.50066102e-01 1.74722120e-01
 1.18508324e-01 8.15714374e-02 7.80973490e-03 9.99997377e-01
 8.98224950e-01 6.53090933e-03 1.01261407e-01 3.64522301e-02
 2.06065159e-02 4.08599436e-01 9.59698930e-02 9.91541207e-01
 4.95125413e-01 9.69866514e-01 9.85935237e-03]
```