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
```

```
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)
```

```
  1/4003 […] - ETA: 41:44 - loss: 1.8021 -
accuracy: 0.3438WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow
compared to the batch time (batch time: 0.0015s vs `on_train_batch_end` time:
0.0078s). Check your callbacks.
4003/4003 [==============================] - 7s 2ms/step - loss: 0.6672 -
accuracy: 0.6066
```

[1]:
```
<tensorflow.python.keras.callbacks.History at 0x7fa22b48cc90>
```

[2]:
```
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([0.4429109 , 0.24186552, 0.33746815, …, 0.41122115, 0.392472  ,
       0.44569722], dtype=float32)
```

[3]:
```
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]:
```
import pandas as pd
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
```

▶ Data pre-loading:

[5]:
```
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]:
```
# 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
print(predicted_prob_true)
```

```
28/28 [==============================] - 1s 19ms/step - loss: 3.1156 - accuracy:
0.6294
[4.5956634e-02 2.5045204e-01 5.1434052e-01 5.0341493e-01 5.2601665e-02
 3.2252628e-02 2.0135791e-04 2.3232634e-01 2.2102144e-02 1.7548123e-01
 7.0456423e-02 1.3825163e-01 2.5755202e-02 2.1712604e-01 3.7076745e-02
 1.4771691e-03 1.2709653e-01 2.1479535e-01 2.9029315e-03 1.5513359e-01
 1.0916419e-01 6.9038771e-02 2.5168591e-04 1.5069677e-01 3.2014742e-01
 4.1736446e-02 1.8013060e-01 7.2635961e-01 4.9539909e-02 1.6717485e-01
 3.5763359e-01 4.2036524e-01 3.4595866e-02 1.0609812e-01 2.1416102e-01
 1.7280167e-01 1.5531802e-01 4.4770289e-02 1.6887575e-01 2.8204811e-01
 1.5904248e-01 3.2595447e-01 3.0528268e-01 1.2362610e-02 2.5197452e-01
 4.8394287e-03 2.4139930e-02 1.5481015e-02 2.2863553e-01 2.0442224e-01
 2.2514185e-01 5.5243977e-06 3.4189633e-01 2.9824379e-01 8.2491070e-02
 2.9127419e-01 1.4131288e-01 3.1110527e-02 3.7589970e-01 3.4595866e-02
 1.3674601e-02 1.7604549e-01 5.2362669e-02 2.9507169e-02 1.7952479e-01
 1.5854310e-02 1.2689959e-01 1.9330798e-01 5.6352358e-02 2.6672539e-01
 7.0547290e-02 1.5474878e-01 8.2864873e-03 2.8222287e-03 2.7167645e-01
 2.8969306e-01 2.0678326e-01 1.7962284e-01 4.3490294e-02 1.7901182e-01
 2.9755443e-01 2.3739902e-02 2.0710088e-01 7.2920546e-02 7.1207143e-02
 3.0904174e-01 1.2200248e-01 2.9367533e-01 3.0490071e-01 2.8705773e-01
 3.2373380e-02]
```