LABORATORIO 6

Semilleros
by DSRP

Ejercicios

En este ejercicio, creará un modelo para predecir cancelaciones de hoteles con un clasificador binario.

```
# Setup plotting
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
# Set Matplotlib defaults
plt.rc('figure', autolayout=True)
plt.rc('axes', labelweight='bold', labelsize='large',
       titleweight='bold', titlesize=18, titlepad=10)
plt.rc('animation', html='html5')

# Setup feedback system
from learntools.core import binder
binder.bind(globals())
from learntools.deep_learning_intro.ex6 import *
```

Semilleros by DSRP

Primero, cargue el conjunto de datos de Cancelaciones de hoteles.

```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer

hotel = pd.read_csv('../input/dl-course-data/hotel.csv')

X = hotel.copy()
y = X.pop('is_canceled')

X['arrival_date_month'] = \
    X['arrival_date_month'].map(
        {'January':1, 'February': 2, 'March':3,
         'April':4, 'May':5, 'June':6, 'July':7,
         'August':8, 'September':9, 'October':10,
         'November':11, 'December':12}
    )
```

```
features_num = [
    "lead_time", "arrival_date_week_number",
    "arrival_date_day_of_month", "stays_in_weekend_nights",
    "stays_in_week_nights", "adults", "children", "babies",
    "is_repeated_guest", "previous_cancellations",
    "previous_bookings_not_canceled", "required_car_parking_spaces",
    "total_of_special_requests", "adr",
]
features_cat = [
    "hotel", "arrival_date_month", "meal",
    "market_segment", "distribution_channel",
    "reserved_room_type", "deposit_type", "customer_type",
]

transformer_num = make_pipeline(
    SimpleImputer(strategy="constant"), # there are a few missing values
    StandardScaler(),
)
transformer_cat = make_pipeline(
    SimpleImputer(strategy="constant", fill_value="NA"),
    OneHotEncoder(handle_unknown='ignore'),
)

preprocessor = make_column_transformer(
    (transformer_num, features_num),
    (transformer_cat, features_cat),
)

# stratify - make sure classes are evenlly represented across splits
X_train, X_valid, y_train, y_valid = \
    train_test_split(X, y, stratify=y, train_size=0.75)

X_train = preprocessor.fit_transform(X_train)
X_valid = preprocessor.transform(X_valid)

input_shape = [X_train.shape[1]]
```

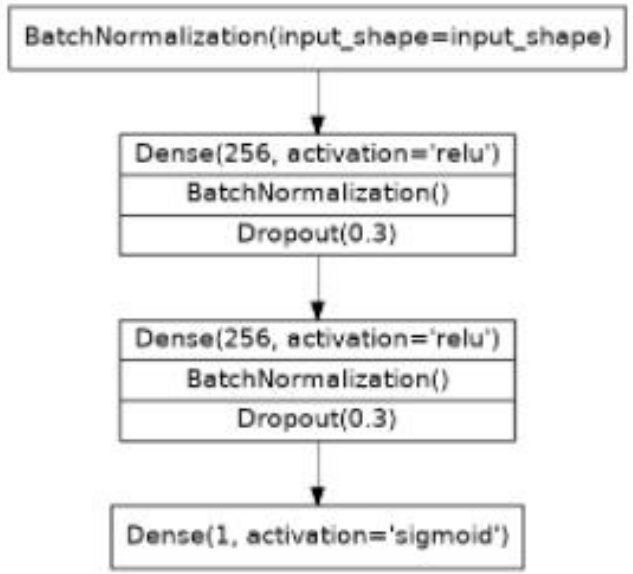
Semilleros by DSRP

Ejercicios

Paso1: Define model

El modelo que usaremos esta vez tendrá tanto la normalización por lotes como las capas de abandono. Para facilitar la lectura, hemos dividido el diagrama en bloques, pero puede definirlo capa por capa como de costumbre.

Defina un modelo con una arquitectura dada por este diagrama:




```
from tensorflow import keras
from tensorflow.keras import layers

# YOUR CODE HERE: define the model given in the diagram
model = keras.Sequential([
     layers.BatchNormalization(input_shape=input_shape),

    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(rate=0.3),

    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(rate=0.3),

    layers.Dense(1, activation='sigmoid'),
])
# Check your answer
q_1.check()
```

Correct

Semilleros by DSRP

Ejercicios

Paso2: Add Optimizer, Loss, and Metric

Ahora compile el modelo con el optimizador de Adam y las versiones binarias de la métrica de precisión y pérdida de entropía cruzada.

```
# YOUR CODE HERE
model.compile(
    optimizer='adam',
     loss='binary_crossentropy',
     metrics=['binary_accuracy'],
)
# Check your answer
q_2.check()
```

Correct

Semilleros by DSRP

Ejercicios

Paso2: Add Optimizer, Loss, and Metric

Finalmente, ejecute esta celda para entrenar el modelo y ver las curvas de aprendizaje

```
early_stopping = keras.callbacks.EarlyStopping(
    patience=5,
    min_delta=0.001,
    restore_best_weights=True,
)
history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    batch_size=512,
    epochs=200,
    callbacks=[early_stopping],
)

history_df = pd.DataFrame(history.history)
history_df.loc[:, ['loss', 'val_loss']].plot(title="Cross-entropy")
history_df.loc[:, ['binary_accuracy', 'val_binary_accuracy']].plot(title="Accuracy")
```

```
Epoch 1/200
175/175 [==============================] - 4s 17ms/step - loss: 0.5328 - binary_accuracy: 0.7443 - val_loss: 0.4310 - val_binary_accuracy: 0.8064
Epoch 2/200
175/175 [==============================] - 2s 14ms/step - loss: 0.4263 - binary_accuracy: 0.7985 - val_loss: 0.4020 - val_binary_accuracy: 0.8147
Epoch 3/200
175/175 [==============================] - 2s 13ms/step - loss: 0.4127 - binary_accuracy: 0.8084 - val_loss: 0.3947 - val_binary_accuracy: 0.8184
Epoch 4/200
```

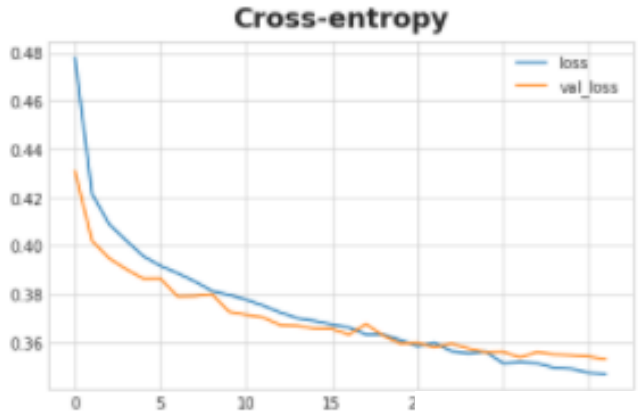
Semilleros by DSRP

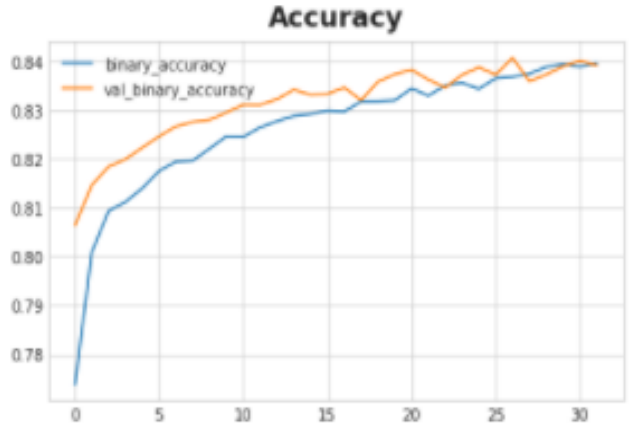
Ejercicios

Paso2: Add Optimizer, Loss, and Metric

Finalmente, ejecute esta celda para entrenar el modelo y ver las curvas de aprendizaje

```
Epoch 21/200
175/175 [==============================] - 2s 13ms/step - loss: 0.3557 - binary_accuracy: 0.8353 - val_loss: 0.3596 - val_binary_accuracy: 0.8383
Epoch 22/200
175/175 [==============================] - 2s 13ms/step - loss: 0.3588 - binary_accuracy: 0.8331 - val_loss: 0.3577 - val_binary_accuracy: 0.8363
Epoch 23/200
175/175 [==============================] - 2s 13ms/step - loss: 0.3503 - binary_accuracy: 0.8382 - val_loss: 0.3593 - val_binary_accuracy: 0.8346
Epoch 24/200
175/175 [==============================] - 2s 14ms/step - loss: 0.3538 - binary_accuracy: 0.8357 - val_loss: 0.3572 - val_binary_accuracy: 0.8372
Epoch 25/200
175/175 [==============================] - 2s 14ms/step - loss: 0.3559 - binary_accuracy: 0.8340 - val_loss: 0.3557 - val_binary_accuracy: 0.8388
Epoch 26/200
175/175 [==============================] - 2s 13ms/step - loss: 0.3495 - binary_accuracy: 0.8367 - val_loss: 0.3559 - val_binary_accuracy: 0.8373
Epoch 27/200
175/175 [==============================] - 2s 14ms/step - loss: 0.3490 - binary_accuracy: 0.8379 - val_loss: 0.3536 - val_binary_accuracy: 0.8407
Epoch 28/200
175/175 [==============================] - 2s 13ms/step - loss: 0.3481 - binary_accuracy: 0.8389 - val_loss: 0.3558 - val_binary_accuracy: 0.8359
Epoch 29/200
175/175 [==============================] - 2s 14ms/step - loss: 0.3464 - binary_accuracy: 0.8399 - val_loss: 0.3548 - val_binary_accuracy: 0.8373
Epoch 30/200
175/175 [==============================] - 2s 14ms/step - loss: 0.3473 - binary_accuracy: 0.8393 - val_loss: 0.3545 - val_binary_accuracy: 0.8390
Epoch 31/200
175/175 [==============================] - 2s 14ms/step - loss: 0.3457 - binary_accuracy: 0.8394 - val_loss: 0.3539 - val_binary_accuracy: 0.8401
Epoch 32/200
175/175 [==============================] - 2s 13ms/step - loss: 0.3442 - binary_accuracy: 0.8411 - val_loss: 0.3527 - val_binary_accuracy: 0.8392
[6]: <AxesSubplot:title={'center':'Accuracy'}>
```





Semilleros by DSRP

Ejercicios

Paso3: Train and evaluate

¿Qué opinas de las curvas de aprendizaje? ¿Parece que el modelo está ajustado o sobreajustado? ¿Fue la pérdida de entropía cruzada un buen sustituto de la precisión?

```
# View the solution (Run this cell to receive credit!)
q_3.check()
```

Correct:

Aunque podemos ver que la pérdida de entrenamiento continúa cayendo, la devolución de llamada de detención temprana evitó cualquier sobreajuste. Además, la precisión aumentó al mismo ritmo que disminuyó la entropía cruzada, por lo que parece que minimizar la entropía cruzada fue un buen sustituto. Considerándolo todo, ¡parece que este entrenamiento fue un éxito!

Semilleros by DSRP