Getting started with neural networks: Classification and regression

Classifying movie reviews: A binary classification example

The IMDB dataset

Loading the IMDB dataset

```
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(
    num_words=10000)
```

```
train_data[0]
```

```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
 670,
 2,
 9,
```

```
 35,
 480,
 284,
 5,
 150,
 4,
 172,
 112,
 167,
 2,
 336,
 385,
 39,
 4,
 172,
 4536,
 1111,
 17,
 546,
 38,
 13,
 447,
 4,
 192,
 50,
 16,
 6,
 147,
 2025,
```

```
train_labels[0]
```

```
1
```

```
max([max(sequence) for sequence in train_data])
```

```
9999
```

Decoding reviews back to text

```
word_index = imdb.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_review = " ".join(
    [reverse_word_index.get(i - 3, "?") for i in train_data[0]])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_w
1646592/1641221 [==============================] - 0s 0us/step
1654784/1641221 [==============================] - 0s 0us/step
```

Preparing the data

Encoding the integer sequences via multi-hot encoding

```
import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        for j in sequence:
            results[i, j] = 1.
    return results
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

```
x_train[0]
```

```
array([0., 1., 1., ..., 0., 0., 0.])
```

```
y_train = np.asarray(train_labels).astype("float32")
y_test = np.asarray(test_labels).astype("float32")
```

Building your model

Model definition

Building the model using 1 Hidden Layer, 64 Hidden unit with tanh activation and mse loss function instead of binary_crossentropy

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

model = keras.Sequential([
    layers.Dense(64, activation="tanh"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
```

Compiling the model

```
model.compile(optimizer="rmsprop",
```

```
              loss="mean_squared_error",
              metrics=["accuracy"])
```

Validating your approach

Setting aside a validation set

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

Training your model

```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 3s 75ms/step - loss: 0.1554 - accuracy: 0.7898
Epoch 2/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0838 - accuracy: 0.9059
Epoch 3/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0638 - accuracy: 0.9261
Epoch 4/20
30/30 [==============================] - 3s 99ms/step - loss: 0.0500 - accuracy: 0.9418
Epoch 5/20
30/30 [==============================] - 2s 72ms/step - loss: 0.0406 - accuracy: 0.9537
Epoch 6/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0352 - accuracy: 0.9596
Epoch 7/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0305 - accuracy: 0.9659
Epoch 8/20
30/30 [==============================] - 2s 58ms/step - loss: 0.0279 - accuracy: 0.9685
Epoch 9/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0223 - accuracy: 0.9767
Epoch 10/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0201 - accuracy: 0.9793
Epoch 11/20
30/30 [==============================] - 2s 58ms/step - loss: 0.0180 - accuracy: 0.9818
Epoch 12/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0160 - accuracy: 0.9838
Epoch 13/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0151 - accuracy: 0.9843
Epoch 14/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0116 - accuracy: 0.9893
Epoch 15/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0108 - accuracy: 0.9893
```

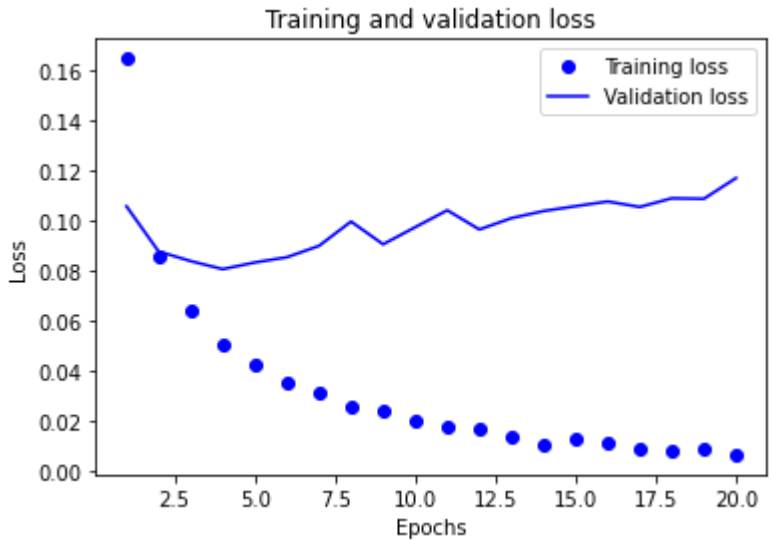
```
Epoch 16/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0108 - accuracy: 0.9890
Epoch 17/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0088 - accuracy: 0.9924
Epoch 18/20
30/30 [==============================] - 2s 58ms/step - loss: 0.0067 - accuracy: 0.9945
Epoch 19/20
30/30 [==============================] - 2s 64ms/step - loss: 0.0070 - accuracy: 0.9938
Epoch 20/20
30/30 [==============================] - 2s 72ms/step - loss: 0.0066 - accuracy: 0.9944
```

```
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

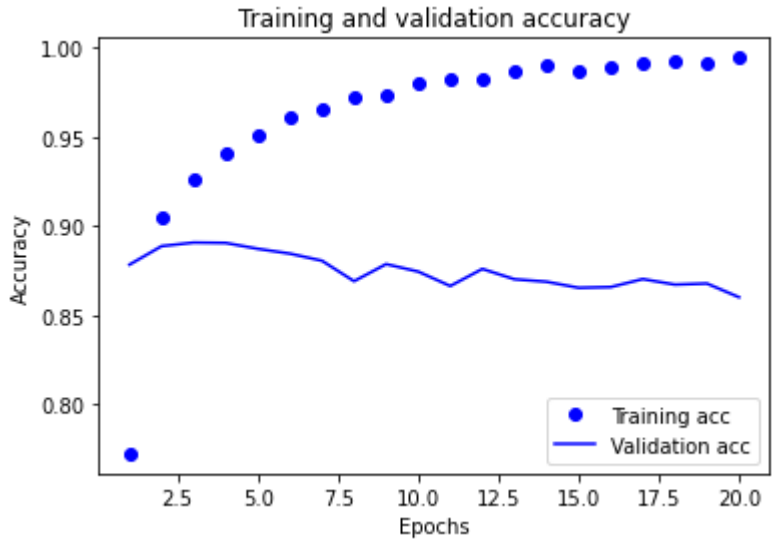
Plotting the training and validation loss

```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



Plotting the training and validation accuracy

```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(64, activation="tanh"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="mean_squared_error",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 [==============================] - 5s 85ms/step - loss: 0.1347 - accuracy: 0.8150
Epoch 2/4
49/49 [==============================] - 3s 58ms/step - loss: 0.0691 - accuracy: 0.9146
Epoch 3/4
49/49 [==============================] - 2s 44ms/step - loss: 0.0537 - accuracy: 0.9334
Epoch 4/4
49/49 [==============================] - 2s 44ms/step - loss: 0.0438 - accuracy: 0.9456
782/782 [==============================] - 2s 3ms/step - loss: 0.1010 - accuracy: 0.8652
```

```
results
```

```
[0.10104542225599289, 0.8651999831199646]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
array([[0.04617223],
       [0.9998853 ],
       [0.63969755],
       ...,
       [0.10897589],
       [0.02025545],
       [0.27655733]], dtype=float32)
```