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]
```

```
      104,
      88,
      4,
      381,
      15,
      297,
      98,
      32,
      2071,
      56,
      26,
      141,
      6,
      194,
      7486,
      18,
      4,
      226,
      22,
      21,
      134,
      476,
      26,
      480,
      5,
      144,
      30,
      5535,
      18,
      51,
      36,
      28,
      224,
      92,
      25,
      104,
      4,
      226,
      65,
      16,
      38,
      1334,
      88,
      12,
      16,
      283,
      5,
      16,
      4472,
      113,
      103,
      32,
      15,
      16,
      5345,
      19,
      178,
      32]
```

```
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_word_index.json
1641221/1641221 [==============================] - 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 and compilation

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

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

```
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
```

Validating the data set

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

Training model

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

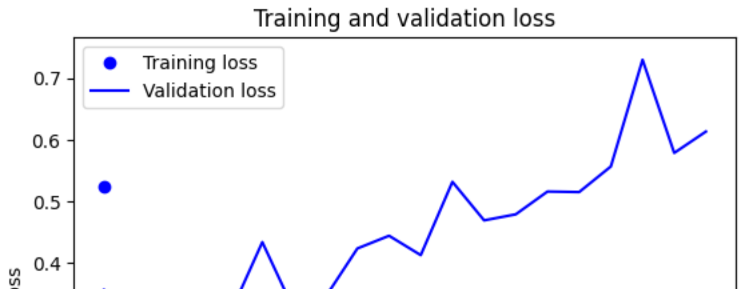
```
Epoch 1/20
30/30 [==============================] - 7s 66ms/step - loss: 0.5228 - accuracy: 0.7411 - val_loss: 0.3559 - val_accuracy: 0.8683
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.3011 - accuracy: 0.8869 - val_loss: 0.2896 - val_accuracy: 0.8884
Epoch 3/20
30/30 [==============================] - 1s 26ms/step - loss: 0.2240 - accuracy: 0.9169 - val_loss: 0.3061 - val_accuracy: 0.8778
Epoch 4/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1794 - accuracy: 0.9331 - val_loss: 0.2776 - val_accuracy: 0.8876
Epoch 5/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1489 - accuracy: 0.9452 - val_loss: 0.3192 - val_accuracy: 0.8724
Epoch 6/20
30/30 [==============================] - 1s 43ms/step - loss: 0.1204 - accuracy: 0.9580 - val_loss: 0.4336 - val_accuracy: 0.8423
Epoch 7/20
30/30 [==============================] - 1s 31ms/step - loss: 0.1027 - accuracy: 0.9636 - val_loss: 0.3231 - val_accuracy: 0.8846
Epoch 8/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0765 - accuracy: 0.9753 - val_loss: 0.3449 - val_accuracy: 0.8796
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0599 - accuracy: 0.9833 - val_loss: 0.4233 - val_accuracy: 0.8748
Epoch 10/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0530 - accuracy: 0.9835 - val_loss: 0.4439 - val_accuracy: 0.8734
Epoch 11/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0414 - accuracy: 0.9877 - val_loss: 0.4125 - val_accuracy: 0.8805
Epoch 12/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0277 - accuracy: 0.9935 - val_loss: 0.5315 - val_accuracy: 0.8503
Epoch 13/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0132 - accuracy: 0.9985 - val_loss: 0.4689 - val_accuracy: 0.8764
Epoch 14/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0358 - accuracy: 0.9890 - val_loss: 0.4788 - val_accuracy: 0.8791
Epoch 15/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0060 - accuracy: 0.9997 - val_loss: 0.5158 - val_accuracy: 0.8784
Epoch 16/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0375 - accuracy: 0.9895 - val_loss: 0.5149 - val_accuracy: 0.8774
Epoch 17/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0037 - accuracy: 0.9999 - val_loss: 0.5565 - val_accuracy: 0.8765
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0117 - accuracy: 0.9969 - val_loss: 0.7298 - val_accuracy: 0.8640
Epoch 19/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0034 - accuracy: 0.9997 - val_loss: 0.5784 - val_accuracy: 0.8761
Epoch 20/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.6132 - val_accuracy: 0.8769
```

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

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

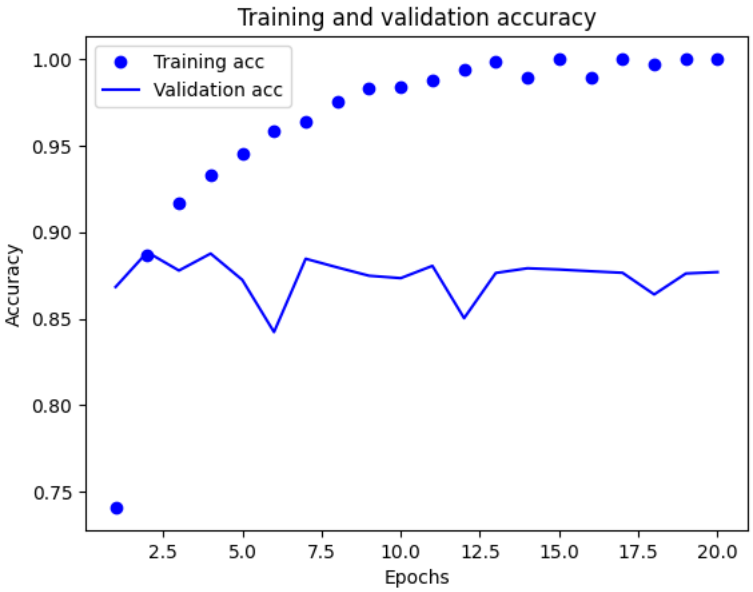
Training and validation loss

```
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()
```



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="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=15, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/15
49/49 [==============================] - 3s 22ms/step - loss: 0.4504 - accuracy: 0.7961
Epoch 2/15
49/49 [==============================] - 1s 19ms/step - loss: 0.2613 - accuracy: 0.8981
Epoch 3/15
49/49 [==============================] - 1s 19ms/step - loss: 0.2100 - accuracy: 0.9181
Epoch 4/15
49/49 [==============================] - 1s 19ms/step - loss: 0.1792 - accuracy: 0.9313
Epoch 5/15
49/49 [==============================] - 1s 14ms/step - loss: 0.1468 - accuracy: 0.9458
Epoch 6/15
```

```
    49/49 [==============================] - 1s 14ms/step - loss: 0.1322 - accuracy: 0.9502
    Epoch 7/15
    49/49 [==============================] - 1s 14ms/step - loss: 0.1113 - accuracy: 0.9594
    Epoch 8/15
    49/49 [==============================] - 1s 15ms/step - loss: 0.0854 - accuracy: 0.9699
    Epoch 9/15
    49/49 [==============================] - 1s 14ms/step - loss: 0.0771 - accuracy: 0.9724
    Epoch 10/15
    49/49 [==============================] - 1s 14ms/step - loss: 0.0487 - accuracy: 0.9861
    Epoch 11/15
    49/49 [==============================] - 1s 14ms/step - loss: 0.0461 - accuracy: 0.9858
    Epoch 12/15
    49/49 [==============================] - 1s 14ms/step - loss: 0.0307 - accuracy: 0.9917
    Epoch 13/15
    49/49 [==============================] - 1s 14ms/step - loss: 0.0336 - accuracy: 0.9880
    Epoch 14/15
    49/49 [==============================] - 1s 16ms/step - loss: 0.0236 - accuracy: 0.9938
    Epoch 15/15
    49/49 [==============================] - 1s 16ms/step - loss: 0.0202 - accuracy: 0.9942
    782/782 [==============================] - 3s 3ms/step - loss: 0.5511 - accuracy: 0.8669
```

```
results
```

```
    [0.5510864853858948, 0.866919994354248]
```