Model Configuration

Configure the Keras model for training.

Chapter Goals:

- Learn how to configure a Keras model for training

A. Configuring for training

When it comes to configuring the model for training, Keras shines with its simplicity. A single call to the `compile` function allows to set up all the training requirements for the model.

The function takes in a single required argument, which is the optimizer to use, e.g. ADAM. A full list of optimizers can be found here. A shorthand way to specify the optimizer is to use a (lowercase) string of the optimizer's name (e.g. `'adam'`, `'sgd'`, `'adagrad'`).

The two main keyword arguments to know are `loss` and `metrics`. The `loss` keyword argument specifies the loss function to use. For binary classification, we set the value to `'binary_crossentropy'`, which is the binary cross-entropy function. For multiclass classification, we set the value to `'categorical_crossentropy'`, which is the multiclass cross-entropy function.

The `metrics` keyword argument takes in a list of strings, representing metrics we want to track during training and evaluation. For classification, we only need to track the model loss (which is tracked by default) and the classification accuracy.

```
model = Sequential()
layer1 = Dense(5, activation='relu', input_dim=4)
model.add(layer1)
layer2 = Dense(1, activation='sigmoid')
model.add(layer2)
model.compile('adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

The code example above creates an MLP model for binary classification and configures it for training. We specified classification accuracy as the metric to track.

Time to code!

The coding exercise will complete the Keras multiclass classification model for training.

To configure the model for training, we need to use the `compile` function. The function sets up the model's loss, optimizer, and evaluation metrics.

For our model, we'll use the ADAM optimizer. Since the model performs multiclass classification, we'll use `'categorical_crossentropy'` for the loss.

The only metric we need to know during training and evaluation is the model's classification accuracy.

Call `self.model.compile` with `'adam'` as the required argument. Use `'categorical_crossentropy'` and `['accuracy']` for the `loss` and `metrics` keyword arguments.

```
model = Sequential()
layer1 = Dense(5, activation='relu', input_dim=2)
model.add(layer1)
layer2 = Dense(5, activation='relu')
model.add(layer2)
layer3 = Dense(3, activation='softmax')
model.add(layer3)
# CODE HERE
```