Experiment 1

Binary Classification with Neural Networks

1.1 Binary Classification of Movie Reviews by Sentiment

In this part of Experiment 1, we'll build a neural network that classifies a movie review as positive or negative. The IMDB dataset, one among the several built-in datasets of Keras, contains movie reviews from the Internet Movie Database. The word **sentiment** refers to the tone, positive or negative, of a review.

Positive reviews are labeled with 1 and negative reviews are labeled with 0. Identifying a review as positive or negative is known as **binary classification** because the output takes on only one of two values. Each review was chosen so that its sentiment is obvious. The dataset has 50,000 reviews, divided evenly between a training dataset and testing dataset, with each dataset itself being divided evenly between positive and negative reviews.

A movie review is a sequence of words. These words must be converted into numbers to enable the calculations done by the neural network. This has already been done by the Keras package maintainers (in other words, the dataset has been preprocessed). Words in a review are indexed by the frequency of their occurrence in the entire dataset. The most frequent word is encoded as the integer 1, the second most frequent word is encoded as the integer 2, etc.

1.1.1 Conventions

Code cells are labeled as Block nn and briefly described in the text cell that immediately precedes them.

Comment lines beginning with `##` indicate places in code cells at which you should enter a comment as described in the procedure.

Write comments so that they are informative to someone with Python coding experience similar to your own.

1.1.2 Importing packages and defining functions

Run the code cells labeled Block 1, Block 2, and Block 3.

The `import` statements are put into two code cells, labeled Block 1 and Block 2, for convenience. TensorFlow and Keras are relatively large packages, so importing them takes a bit more time than usual (a few seconds). Importing them only when necessary (probably just once, at the start of the session) is an efficiency. If you decide to import an additional package, it is best to put the `import` statement in the first code cell because importing those packages takes relatively little time.

Insert a code cell above Block 3. Enter the statement
`dir(imdb)`
in this cell. Run the cell. Add statements as needed to help you understand the `imdb` module, rerunning the cell after each addition. `help` is especially useful when becoming acquainted with a new module. Clean up the cell by removing statements that turned out to be less helpful. Describe what you learned about the IMDB dataset and how to work with it from the output of `dir` in a comment following the last statement (`dir` if you didn't add any statements, otherwise the last statement you added).

Functions that we want to be available throughout the session are defined in Block 3. Add a comment above the `return` statement in `decode_review` that explains how it translates an encoded review back to something close to the original text (the translation will not be exact if uncommon words appeared in the original review).

1.1.3 Loading the IMDB dataset

Run the code cell labeled Block 4.

Insert a code cell above Block 5. Write a code snippet in this cell that displays the first review of the training dataset; its data type; the label used to classify it as positive or negative; and the number of reviews in the training dataset. Write a comment following the last executable statement of your code snippet that answers the questions

- Does the review appear as expected based on your reading of the background information?
- Is the review positive or negative? Briefly explain how you reached your conclusion.
- Does the training set contain the advertised number of reviews?

1.1.4 Loading the word to rank encoding dictionary and creating its rank to word companion

Run the code cell labeled Block 5.

Insert a code cell above Block 6. Write a code snippet in this cell that displays the number of words in the word to rank dictionary and the ten most frequent words. Write a comment following the last executable statement of your code snippet that answers the questions

- Only the 10,000 most frequent words were retained when the dataset was loaded. What fraction of the total number of words in the dataset is this?
- What are the ten most frequent words? Are you surprised by any of the words in the list?

1.1.5 Reconstructing a review from the IMDB dataset

Run the code cell labeled Block 6.

Read the reconstructed original review. Do you agree with its label as positive or negative? Why or why not? Write your answer as a comment at the end of the code cell.

1.1.6 Recoding the training and test datasets

The training and test datasets are stored as 1-D NumPy arrays, with each element being a list of word indices. The length of one of these lists is the number of words in the review. Because the reviews have many different lengths, the lists have many different lengths. This data structure is not suitable as the input layer to a neural network.

One way to reformat the encoded reviews so that they will function as the input layer is to mark each word that occurs in a review with a 1 and all the other words with a 0. This is known as one-hot encoding. As an example, a two-word review made up of the 6th and 3rd most frequent words is represented by $[6, 3]$ in the original encoding and by $[0, 0, 1, 0, 0, 1, 0, 0, \ldots, 0]$ after recoding as one-hot. The length of the new encoding is the number of words that are indexed. The original 1-D NumPy array becomes a 2-D array having as many rows as there are reviews and as many columns as the number of words that are indexed.

To recode the training and test datasets, run the code cell labeled Block 7.

Insert a code cell above Block 8. Enter a code snippet that checks the size of the input array against the expectations spelled out above. Display the results of this check using a format something like `The row dimension of the input array, NNNN, is correct.` or `The row dimension of the input array, NNNN, is incorrect.`, where `NNNN` is the number of rows you found the input array to have. Include an analogous statement for the column dimension.

1.1.7 Building and training the model

Each recoded review is a vector (a 1-D array, as a row of a 2-D array). Each label is a scalar (0 or 1). A model that suits this situation is a sequence of densely connected hidden layers with ReLU activation for all but the last hidden layer and sigmoid activation for the last hidden layer. Figure 1.1 shows graphs of these activation functions.

Designing the model requires choosing the number of hidden layers and the number of nodes in each of the hidden layers. There are guidelines for making these choices, which we will discuss in the coming weeks. As a starting point, I have adopted the recommendations of Chollet, whose code in *Deep Learning with Python* is the inspiration for this experiment, in the first statement of Block 8.

The `compile` method of the model, implemented in the second statement of Block 8, completes its construction. The loss function, specified by the `loss` argument, measures the difference between the target values of the output (this is known as the **ground truth**) and the current predicted output values. Minimizing the loss function is the goal. The strategy used to achieve this is specified by the `optimizer` argument. The choice of optimization algorithm and of loss function is guided by the nature of the input and output data. We will develop guidelines for making these choices in the coming weeks.

It is standard practice to train the model on a fraction of the entire training data set and to validate the trained model on the remaining fraction of the training data set. The IMDB database is randomized, so this split can be made simply by slicing the training dataset. This is done in Block 9.

The `fit` method of the model is the routine in which the parameters (the weights) of the neural network are refined. Refinement occurs as the loss function decreases. The number of refinement steps is set by the `epochs` argument. The dataset is processed in subsets whose size is set by the `batch_size` argument. Providing validation data via the `validation_data` argument enables checking the quality of the model during each epoch.

Run the code cells labeled Block 8, Block 9, and Block 10 to build and train the model. Estimate the average time elapsed during an epoch. Record this time in a comment at the end of the code cell of Block 10.

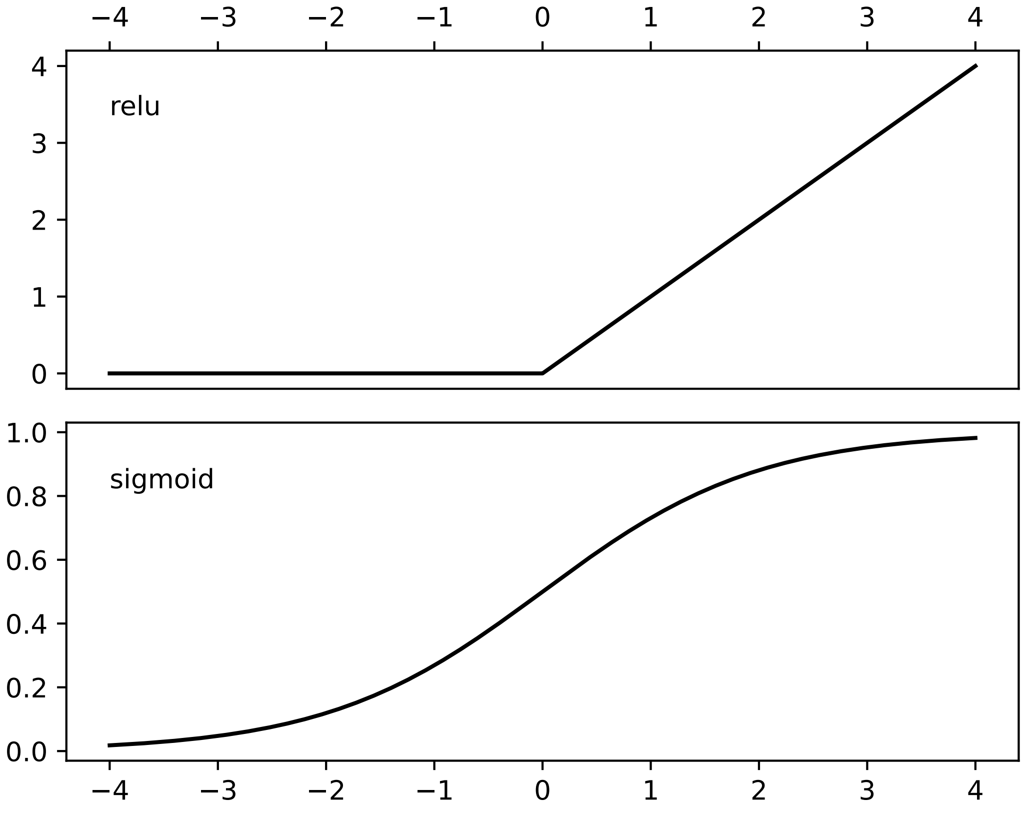

Figure 1.1: ReLU and sigmoid activation functions. ReLU, which stands for rectified linear unit, flattens negative input by setting it to zero and tracks positive input 1 : 1 (the line segment for $x \geq 0$ is $y = x$). Sigmoid, which means curved in two directions, sort of like the letter S, converts any input value to an output value between 0 and 1. This output can be made binary by mapping all input values giving an output value less than 0.5 to 0 and all input values giving an output value greater than 0.5 to 1.

1.1.8 Visualizing the training history

The `fit` method of a Keras model creates a History object with a history dictionary whose keys are named following the metrics for the training and validation stages (e.g., `loss`) and whose values are lists of the metrics after each epoch. These values can be plotted versus the epoch number to visualize the progress of the fit.

Run the code cell labeled Block 11.

Examine the trends in loss vs epoch number and accuracy vs epoch number displayed in the figure.

Answer the questions in the accompanying list in a text cell inserted above Block 12.

- Which can be described as a direct relationship? Which can be described as an inverse relationship? Which are neither a direct relationship nor an inverse relationship?
- Drawing on your experience from Machine Learning, comment on the divergence between the validation loss and the training loss.
- Based on the figure, estimate the number of epochs that will result in the best model. Briefly discuss how you arrived at your estimate, included how you defined "best".

1.1.9 Retraining the model, from the ground up

Repeating the steps of Block 8 and Block 10 creates a new model. No memory of the previous training cycles is retained. Having used the validation data to determine the optimal number of epochs, we can now use the entire training dataset at this stage.

Enter the number of epochs from your answer to the third item in the previous subsection in the assignment on the first statement of Block 12 and run the code.

1.1.10 Evaluating the retrained model

The `evaluate` method of a Keras model uses the model to make predictions using the test dataset, returning the accuracy and the loss.

Run Block 13. Record the accuracy and loss in a comment at the end of the code cell of Block 13.

1.1.11 Making predictions using the retrained model

The `evaluate` method of a Keras model uses the model to predict the sentiments of the reviews in the test dataset. The returned sentiments are the values of the sigmoid function, so they are floating point numbers between 0 and 1. A number close to 1 is interpreted as indicating a positive review, and a number close to 0 is interpreted as indicating a negative review. That the model developer decides how close is close enough will be a recurring theme. As a model developer, you have considerable latitude in making this decision, but you do need to be prepared to providek a rationale.

Run Block 14. Identify a predicted sentiment that you would label as positive, a predicted sentiment that you would label as negative, and a predicted sentiment that you would label as indeterminate.

Using Block 6 as a guide, add code following the `print` statement to display each of the reviews. Say whether you agree with the predicted sentiments in a comment following the code.