Normalizing Data

Learn about data normalization and implement a normalization function.

Chapter Goals:

- Learn how to apply L2 normalization to data

L2 normalization

So far, each of the scaling techniques we've used has been applied to the data features (i.e. columns). However, in certain cases we want to scale the individual data observations (i.e. rows). For instance, when clustering data we need to apply L2 normalization to each row, in order to calculate cosine similarity scores. The **Clustering** section will cover data clustering and cosine similarities in greater depth.

L2 normalization applied to a particular row of a data array will divide each value in that row by the row's L2 norm. In general terms, the L2 norm of a row is just the square root of the sum of squared values for the row.

$$X = [x_1, x_2, ..., x_m]$$

$$X_{L2} = \left[\frac{x_1}{\ell}, \frac{x_2}{\ell}, ..., \frac{x_m}{\ell}\right], \text{where } \ell = \sqrt{\sum_{i=1}^{m} x_i^2}$$

The above formula demonstrates L2 normalization applied to row X to obtain the normalized row of values, X_{L2}.

In scikit-learn, the transformer module that implements L2 normalization is the `Normalizer`.

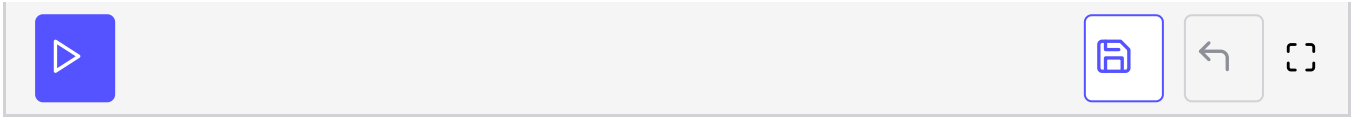
The code below shows how to use the `Normalizer`.

```
# predefined data
```

```
print('{}\n'.format(repr(data)))

from sklearn.preprocessing import Normalizer

normalizer = Normalizer()
transformed = normalizer.fit_transform(data)
print('{}\n'.format(repr(transformed)))
```



Time to Code!

The coding exercise in this chapter uses `Normalizer` (imported in backend) to complete the `normalize_data` function.

The function will apply L2 normalization to the input NumPy array, `data`.

Set `normalizer` equal to `Normalizer` initialized without any parameters.

Set `norm_data` equal to `normalizer.fit_transform` applied with `data` as the only argument. Then return `norm_data`.

```
def normalize_data(data):
  # CODE HERE
  pass
```