

# Data Range

Create a function to compress data into a specific range of values.

## Chapter Goals:

- Learn how to compress data values to a specified range

### A. Range scaling

Apart from standardizing data, we can also scale data by compressing it into a fixed range. One of the biggest use cases for this is compressing data into the range  $[0, 1]$ . This allows us to view the data in terms of proportions, or percentages, based on the minimum and maximum values in the data.

The formula for scaling based on a range is a two-step process. For a given data value,  $x$ , we first compute the proportion of the value with respect to the min and max of the data  $d_{\min}$  and  $d_{\max}$ , respectively).

$$x_{prop} = \frac{x - d_{min}}{d_{max} - d_{min}}$$

The formula above computes the proportion of the data value,  $x_{prop}$ . Note that this only works if not all the data values are the same (i.e.  $d_{\max} \neq d_{\min}$ ).

We then use the proportion of the value to scale to the specified range,  $[r_{\min}, r_{\max}]$ . The formula below calculates the new scaled value,  $x_{scale}$ .

$$x_{scale} = x_{prop} \cdot (r_{max} - r_{min}) + r_{min}$$

### B. Range compression in scikit-learn

The scikit-learn library provides a variety of *transformers*, modules that perform transformations on data. While in the previous chapter we used a single function, `scale`, to perform the data standardization, the remaining chapters will focus on using these transformer modules.

The `MinMaxScaler` ([https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler)

[learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler)) transformer performs the range compression using the previous formula. Specifically, it scales each feature (column) of the data to a given range (where the default range is `[0, 1]`).

The code below shows how to use the `MinMaxScaler` (with the default range and a custom range).

The `MinMaxScaler` object contains a function called `fit_transform`, which allows it to take in the input data array and then output the scaled data. The function is a combination of the object's `fit` and `transform` functions, where the former takes in an input data array and the latter transforms a (possibly different) array based on the data from the input to the `fit` function.

```
1 # predefined data
2 print('{}\n'.format(repr(data)))
3
4 from sklearn.preprocessing import MinMaxScaler
5 default_scaler = MinMaxScaler() # the default range is [0,1]
6 transformed = default_scaler.fit_transform(data)
7 print('{}\n'.format(repr(transformed)))
8
9 custom_scaler = MinMaxScaler(feature_range=(-2, 3))
10 transformed = custom_scaler.fit_transform(data)
11 print('{}\n'.format(repr(transformed)))
```



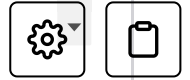
Output

1.460s

```
array([[ 1.2,  3.2],
       [-0.3, -1.2],
       [ 6.5, 10.1],
       [ 2.2, -8.4]])
```

```
array([[0.22058824, 0.62702703],
       [0.          , 0.38918919],
```

```
[1.          , 1.          ],  
[0.36764706, 0.          ]])
```



Now let's run the `fit` and `transform` functions separately and compare them with the `fit_transform` function. `fit` takes in an input data array and `transform` transforms a (possibly different) array based on the data from the input to the `fit` function.

```
1 # predefined new_data  
2 print('{}\n'.format(repr(new_data)))  
3  
4 from sklearn.preprocessing import MinMaxScaler  
5 default_scaler = MinMaxScaler() # the default range is [0,1]  
6 transformed = default_scaler.fit_transform(new_data)  
7 print('{}\n'.format(repr(transformed)))  
8  
9 default_scaler = MinMaxScaler() # new instance  
10 default_scaler.fit(data) # different data value fit  
11 transformed = default_scaler.transform(new_data)  
12 print('{}\n'.format(repr(transformed)))
```



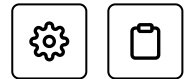
Output

0.802s

```
array([[ 1.2, -0.5],  
       [ 5.3,  2.3],  
       [-3.3,  4.1]])  
  
array([[0.52325581, 0.          ],  
       [1.          , 0.60869565],  
       [0.          , 1.          ]])  
  
array([[ 0.22058824,  0.42702703],
```

The code above scales the `new_data` array to the range `[0, 1]`, based on the (column-wise) minimum and maximum values from the `data` array in the original code example.

## Time to Code!



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

The function will compress the input NumPy array, `data`, into the range given by `value_range`.

**Set `min_max_scaler` equal to `MinMaxScaler` initialized with `value_range` for the `feature_range` keyword argument.**

**Set `scaled_data` equal to `min_max_scaler.fit_transform` applied with `data` as the only argument. Then return `scaled_data`.**

```
1 def ranged_data(data, value_range):
2     # CODE HERE
3     pass
```



Solution



```
1 def ranged_data(data, value_range):
2     min_max_scaler = MinMaxScaler(feature_range=value_range)
3     scaled_data = min_max_scaler.fit_transform(data)
4     return scaled_data
5
6
7
```

← Back

Standardizing Data

Next →

Robust Scaling

☒ Mark as Completed

32% completed, meet the [criteria](#) and claim your course certificate!

Buy Certificate



Report an  
Issue



Ask a Question

([https://discuss.educative.io/tag/data-range\\_\\_data-preprocessing-with-scikit-learn\\_\\_machine-learning-for-software-engineers](https://discuss.educative.io/tag/data-range__data-preprocessing-with-scikit-learn__machine-learning-for-software-engineers))

