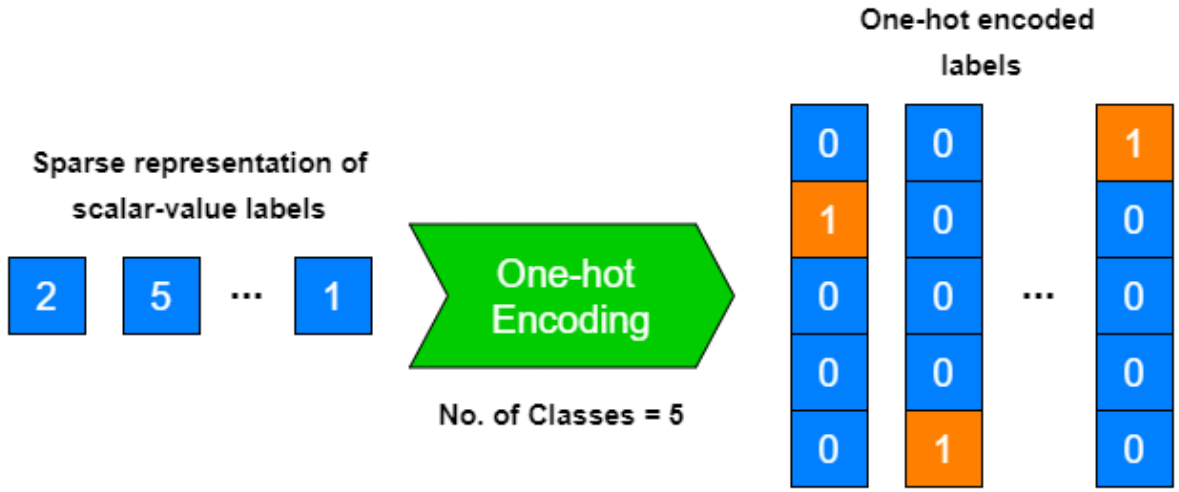


(Cover image by author, made with draw.io)

We need to convert scalar-value labels into a ***one-hot vector*** before using them in deep learning models.

This is required for multiclass classification models that output probabilities per class when using the ***categorical_crossentropy*** loss function.

Sparse scalar representation

The values in the label column are usually represented as sparse scalars (i.e. in single-digit format). For example, the training and test labels in the MNIST digits dataset are represented in single-digit format ranging from 0 to 9. Each digit represents a class label.

```
Training labels
[5 0 4 ... 5 6 8]
(60000,)
<class 'numpy.ndarray'>
----------------------
Test labels
[7 2 1 ... 4 5 6]
(10000,)
<class 'numpy.ndarray'>
```

(Image by author)

The training and test labels are in two separate one-dimensional vectors.

One-hot representation

As I explained at the beginning of this article, sparse scalar representation is not suitable for multiclass classification models that output probabilities per class. So, it is necessary to perform one-hot encoding for scalar-value labels before using them in deep learning models.

After performing one-hot encoding, we get a one-hot vector.

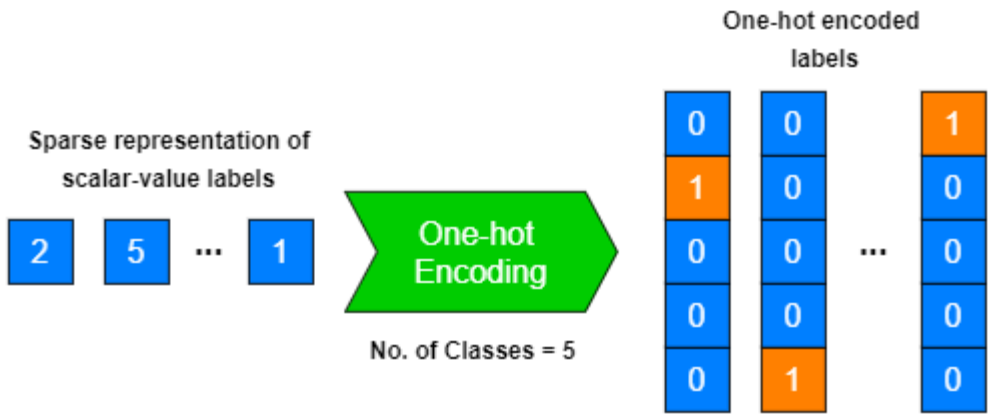
In one-hot encoding, we need an ***n-element vector*** (**n** is the number of classes) for each scalar-value label. All elements in the vector are zeros (denote 0% probability) except the one that we use to mark the index of the scalar value. For example, the one-hot representation of the scalar-value 5 can be repeated as in the following when the number of classes is 10.

```
#5
[0,0,0,0,0,1,0,0,0,0]
```

The value 1 at the fifth index (indices is zero-based) denotes 100% probability for the label value at that index. That's digit 5.

We get that type of vector for each scalar-value label and stack them vertically to get the final one-hot vector.

Here is an example of a one-hot vector of three scalar-value labels when the number of classes is 5.



(Image by author, made with draw.io)

After performing one-hot encoding, the training and test labels in the MNIST digits dataset look as follows.

```
Training labels
[[0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]]
(60000, 10)
<class 'numpy.ndarray'>
----------------------
Test labels
[[0. 0. 0. ... 1. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
(10000, 10)
<class 'numpy.ndarray'>
```

(Image by author)

Now, the training and test labels are in two separate two-dimensional matrices.