Video

Bookmark this page

Feature Representation

Start of transcript. Skip to the end.

(Caption will be displayed when you start playing the video.)

0:00 / 7:41 1.25x

We talked about preparing categorical data for machine learning.

You can convert an ordinal feature into a discrete, numeric one by making sure you preserve the numeric encoding in its order.

As for nominal features, you saw that there were two routes for that.

You can either convert it into a

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Pure Textual Features

If you are trying to "featurize" a body of text such as a webpage, a tweet, a passage from a newspaper, an entire book, or a PDF document, creating a corpus of words and counting their frequency is an extremely powerful encoding tool. This is also known as the *Bag of Words* model, implemented with the CountVectorizer() method in SciKit-Learn. Even though the

grammar of your sentences and their word-order are complete discarded, this model has accomplished some pretty amazing things, such as correctly identifying J.K. Rowling's writing from a blind line up of authors.

```
>>> from sklearn.feature_extraction.text import CountVectorizer

>>> corpus = [
...   "Authman ran faster than Harry because he is an athlete.",
...   "Authman and Harry ran faster and faster.",
... ]

>>> bow = CountVectorizer()
>>> X = bow.fit_transform(corpus) # Sparse Matrix

>>> bow.get_feature_names()
['an', 'and', 'athlete', 'authman', 'because', 'faster', 'harry', 'he', 'is', 'ran',
'than']

>>> X.toarray()
[[1 0 1 1 1 1 1 1 1 1 1]
 [0 2 0 1 0 2 1 0 0 1 0]]
```

Some new points of interest. X is not the regular [n_samples, n_features] dataframe you are familiar with. Rather, it is a SciPy compressed, sparse, row matrix. SciPy is a library of mathematical algorithms and convenience functions that further extend NumPy. The reason X is now a sparse matrix instead of a classical dataframe is because even with this small example of two sentences, 11 features got created. The average English speaker knows around 8000 unique words. If each sentence were an 8000-sized vector sample in your dataframe, consisting mostly of 0's, it would be a poor use of memory.

To avoid this, SciPy implements their sparse matrices like Python implements its dictionaries: only the keys that have a value get stored, and everything else is assumed to be empty. You can always convert it to a regular Python list by using the .toarray() method, but this converts it to a dense array, which might not be desirable due to memory reasons. To use your compressed, spare, row matrix in Pandas, you're going to want to convert it to a Pandas SparseDataFrame. More notes on that in the Dive Deeper section.

The bag of words model has other configurable parameters, such as considering the order of words. In such implementations, pairs or tuples of successive words are used to build the corpus instead of individual words:

```
>>> bow.get_feature_names()
['authman ran', 'ran faster', 'faster than', 'than harry', 'harry because', 'because he',
'he is', 'is an', 'an athlete', 'authman and', 'and harry', 'harry ran', 'faster and',
'and faster']
```

Another parameter is to have it use frequencies ant not counts. This is useful when you have documents of different lengths, so to allow direct comparisons even though the raw counts for the longer document would of course be higher. Dive deeper into the feature extraction section of SciKit-Learn's documentation to learn more about how you can best represent your textual features!

English