 1 help

scikit multi label classification

I am trying to classify data into four different labels. The training data looks something like:

```
[
  [[0.1 , 0.6, 0.0, 0.3], 1, 10, 0, 0, 0],
  [[0.7 , 0.3, 0.0, 0.0], 0, 7, 22, 0, 0],
  [[0.0 , 0.0, 0.6, 0.4], 0, 0, 6, 0, 20],
  ...
]
```

Each row is an instance. First column contains the labels. Columns `2..n` contain the features.

I am trying to understand how to train a multilabel classifier using SciKit OneVsRestClassifier but I just don't get how I must proceed in the code.

I have been able to do single-label classification for this dataset replacing the first column with a single value (eg. the first instance would map completely to the second label) but I would like the more nuanced multi-label output. This is what I use for the single label classifier (assuming above dataset):

```
labels = [1,0,3, ...]
data = [[1, 10, 0, 0, 0], [0, 7, 22, 0, 0], [0, 0, 6, 0, 20], ...]
clf = svm.SVC(kernel='poly')
clf.fit(data, labels)
```

Any idea how to convert this to multi-labels?

Thanks!

classification scikit-learn multilabel

asked Oct 3 '14 at 21:10

mga
113 1 4

You may be mixing up multi-class with multi-output. What are your target values in this example? It looks like you'd want to predict on 4 non-discrete outputs for each row. Where as a multi-class scikit-learn model just predicts one class (out of many) for each observation. – eric chiang Oct 4 '14 at 13:51

@ericchiang I am classifying books by "appeal" (eg: story/character/setting-driven). These appeals are not mutually exclusive and are subjective. I want to avoid putting every book in a single bucket. – mga Oct 5 '14 at 22:29

One vs Rest and One vs One "meta" classifiers are (usually) used to help predictors which would normally only be able to differentiate between two classes, like a logistic regression, to allow them to be able to assign a single class out of many. However, the final result is still only a classifier that produces a single label. If you have 4 labels, they will still only predict 0, 1, 2 or 3 (exclusively), not a combination of the labels you feed in. For instance, the result will never be both 0 and 1. – eric chiang Oct 6 '14 at 3:58

1 Answer

The Multi-label algorithm accepts a binary mask over multiple labels. So, for example, you could do something like this:

```
data = [
        [[0.1 , 0.6, 0.0, 0.3], 1, 10, 0, 0, 0],
        [[0.7 , 0.3, 0.0, 0.0], 0, 7, 22, 0, 0],
        [[0.0 , 0.0, 0.6, 0.4], 0, 0, 6, 0, 20],
        #...
       ]

X = np.array([d[1:] for d in data])
yvalues = np.array([d[0] for d in data])

# Create a binary array marking values as True or False
from sklearn.preprocessing import MultiLabelBinarizer
Y = MultiLabelBinarizer().fit_transform(yvalues)

clf = OneVsRestClassifier(SVC(kernel='poly'))
clf.fit(X, Y)
clf.predict(X)  # predict on a new X
```

The result for each prediction will be an array of 0s and 1s marking which class labels apply to each row input sample.

Given your data, though, I'm not sure this is what you want to do. For example, the third point has zero listed twice, which makes me think that you're not predicting multiple labels in an unordered `OneVsRest` manner, but actually predicting multiple ordered columns of labels: in

that case, it might make sense to do a separate classification for each, e.g.

```
X = np.array([d[1:] for d in data])
Y = np.array([d[0] for d in data])
clfs = [SVC().fit(X, Y[:, i]) for i in range(Y.shape[1])]
Ypred = np.array([clf.predict(X) for clf in clfs]).T
```

With other classifiers, such as `RandomForestClassifier` , you can do this column-by-column prediction in one operation: e.g.

```
X = np.array([d[1:] for d in data])
Y = np.array([d[0] for d in data])
RandomForestClassifier().fit(X, Y).predict(X)
```

Of course, the array passed to `predict` should be on something different than the array passed to `fit` , but hopefully this makes the distinction clear.

edited Oct 3 '14 at 22:41

answered Oct 3 '14 at 22:27

jakevdp

146 3

to clarify the data: these are books. first column is the "appeal" label of the book (ie. story/character/setting/language-driven). second-n columns are counts for a given adjective in its reviews. so that's why some labels have 0. i think the second example solves my problem: separate classification for each. thanks – mga Oct 6 '14 at 16:27

1 My comment wasn't about there being zeros, it was about the labels being multiple ordered columns. One-Vs-Rest treats multi-labels as unordered sets. For example, the label [0,0,4,5] would be considered identical to the label [5,0,5,4], and identical to [0, 4, 5]: all that matters is whether a label is present or absent. Given what you've said, I think you're actually interested in the ordered columns themselves, which is why label binarization may not be the right solution. – jakevdp Oct 6 '14 at 17:20

Got it. Understood. Thanks – mga Oct 6 '14 at 18:38