Machine Learning

| Multilabel Classification

Prof. Dr. Mostafa Elhosseini

https://youtube.com/@ProfElhosseiniTechVerse

Agenda

- Multilabel Classification

Intro

- Until now each instance has always been assigned to just one class.
- In some cases you may want your classifier to output multiple classes for each instance.
- For example, consider a face-recognition classifier: what should it do if it recognizes several people on the same picture? Of course it should attach one label per person it recognizes.
- Say the classifier has been trained to recognize three faces, Alice, Bob, and Charlie; then when it is shown a picture of Alice and Charlie, it should output [1, 0, 1] (meaning “Alice yes, Bob no, Charlie yes”).
- Such a classification system that outputs multiple binary labels is called a multilabel classification system

Multilabel classification

- **Multilabel classification**—A classification task where **each input** sample can be **assigned multiple labels**. For instance, a given image may contain both a cat and a dog and should be annotated both with the “cat” label and the “dog” label. The **number of labels** per image is usually **variable**.
- Each label is **binary**, indicating the presence or absence of something

Multilabel classification

- A value of 1 indicates the presence of the label, while 0 indicates its absence. For example, in a text categorization problem with labels "politics," "economy," and "sports," an article classified as both "politics" and "economy" would have a label vector of [1, 1, 0].

Multilabel classification

- Image annotation: An image can be tagged with multiple labels, such as "beach," "sunset," "people," and "sand" at the same time.
- Medical diagnosis: A patient can be diagnosed with multiple medical conditions concurrently, such as "diabetes," "hypertension," and "obesity."
- Music genre classification: A song can belong to multiple genres, like "rock," "alternative," and "indie" simultaneously.

Multilabel classification

- Classify **Reuters newswires** into **46 mutually exclusive topics**. Because we have many classes, this problem is an instance of **multiclass classification**, and because each data point should be classified into **only one category**, the problem is more specifically an instance of **single-label multiclass classification**.
- If each data point could belong to **multiple categories** (in this case, topics), we'd be facing a **multilabel multiclass classification problem**.

Multilabel classification

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

y_train_large = (y_train >= '7')
y_train_odd = (y_train.astype('int8') % 2 == 1)
y_multilabel = np.c_[y_train_large, y_train_odd]

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_multilabel)
```

- This code creates a y_multilabel array containing two target labels for each digit image
- the first indicates whether or not the digit is large (7, 8, or 9), and the second indicates whether or not it is odd
- Then the code creates a KNeighborsClassifier instance, which supports multilabel classification (not all classifiers do)

Multilabel classification

```
knn_clf.predict([some_digit])
```

```
array([[False,  True]])
```

- The digit 5 is indeed not large (False) and odd (True).

Evaluation

- There are many ways to evaluate a multilabel classifier, and selecting the right metric really depends on your project.
- For example, one approach is to **measure the F1 score** for each individual label (or any other binary classifier metric discussed earlier), then simply compute the average score

```
y_train_knn_pred = cross_val_predict(knn_clf, X_train, y_multilabel, cv=3)
f1_score(y_multilabel, y_train_knn_pred, average="macro")
```

```
0.976410265560605
```

- This assumes that all labels are equally important, which may not be the case.

Evaluation

- In particular, if you have many more pictures of Alice than of Bob or Charlie, you may want to give more weight to the classifier's score on pictures of Alice.
- One simple option is to give each label a weight equal to its support (i.e., the number of instances with that target label).
- To do this, simply set **average="weighted"** in the preceding code

```
f1_score(y_multilabel, y_train_knn_pred, average="weighted")
```

```
0.9778357403921755
```

- Negligible performance improvement - because the classes are already pretty well balanced

Note:

- If you wish to use a classifier that does not natively support multilabel classification, such as SVC, one possible strategy is to train one model per label.
- However, this strategy may have a hard time capturing the dependencies between the labels
- Often, there is **some relationship or correlation** between the outputs, and models may **need to capture these relationships** to perform effectively. For example, predicting multiple diseases that a patient might have based on their symptoms where **diseases can be correlated**.

Note:

- Multilabel classification assumes a **level of independence** among the labels; each label represents a binary decision, though in practice, labels can still be correlated.
- In **multioutput** classification, there can be a higher degree of interdependence among outputs, as each output might influence the other (e.g., outputting a category might determine the range of another numeric output).
- See Next Lecture

Next Lecture

- The terms "multilabel" and "**multioutput**" classification are often used interchangeably in machine learning, but they can represent slightly different scenarios.

Multilabel Classification

- x

Multilabel classification

- Reuters dataset, a set of short newswires and their topics, published by Reuters in 1986. It's a simple, widely used toy dataset for text classification. There are 46 different topics; some topics are more represented than others, but each topic has at least 10 examples in the training set.
- Like IMDB and MNIST, the Reuters dataset comes packaged as part of Keras