test_rocket

January 31, 2023

0.1 Advanced Topics in ML & DM

0.2 Report Time Series Classification

0.2.1 Nicolas Rojas

ROCKET (for RandOm Convolutional KErnel Transform) is a time series classification approach that uses random convolutional kernels so as to transforme a time series into features useful for a linear classifier.

This work shows a comparison beetween ROCKET aproach and other algorithms for Time Series Classification as SAST, HIVE-COTE, ST and Catch22. All these methods are implemented in python through the sktime framework.

In order to compare results obtained for the different aproaches, the algorithms were applied to the Chinatown Dataset which contains pedestrian count in Chinatown-Swanston St North for 12 months of the year 2017. Additionally, classes, for this dataset, are based on whether data are from a normal day or a weekend day. - Class 1: Weekend - Class 2: Weekday.

The implementation of the different methods were performed locally using an Intel Core i7-6820HQ CPU @ 2.70GHz processor.

0.2.2 Using ROCKET on Chinatown Dataset:

Firstly, in order to apply ROCKET, to the Chinatown dataset (which contains 20 objects of length 24), the libraries load_UCR_UEA_dataset and RocketClassifier are imported.

[ ]:
```
from sktime.datasets import load_UCR_UEA_dataset
from sktime.classification.kernel_based import RocketClassifier
```

[ ]:
```
X_train, y_train = load_UCR_UEA_dataset(name="Chinatown",extract_path='data',␣
↪split="train",return_type="numpy2d")
X_test, y_test = load_UCR_UEA_dataset(name="Chinatown",extract_path='data',␣
↪split="test",return_type="numpy2d")
print("shape X_train DS:", X_train.shape)
```

```
shape X_train DS: (20, 24)
```

An instance of RocketClassifier is created (with 10.000 kernels) and fitted with the training dataset. The fitting step takes a duration of aprox. 7s and the accuracy of the model (#correct_preds / #all_preds) is 0.9825, using the testing dataset.

[ ]:
```
rocket= RocketClassifier(num_kernels=10_000)
rocket.fit(X_train,y_train)
```

[ ]:
```
RocketClassifier()
```

[ ]:
```
rocket.score(X_test,y_test)
```

[ ]:
```
0.9825072886297376
```

0.2.3 Using SAST on Chinatown Dataset:

Additionally, for the Chinatown dataset the SAST algorithm was applied. This implementation was cloned from the following URL: https://github.com/frankl1/sast

[ ]:
```
from sast.utils import *
from sast.sast import *
```

[ ]:
```
min_shp_length = 3
max_shp_length = X_train.shape[1]

candidate_lengths = np.arange(min_shp_length, max_shp_length+1)
nb_inst_per_class = 1
print('candidate_lengths:',candidate_lengths)
```

```
candidate_lengths: [ 3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24]
```

SAST was used in combination with a ridge classifier obtaining an accuracy of 0.9 approximately. Furthermore the time to create the Ridge Classifier instance, the SAST instance and fitting the model was of about 1s.

[ ]:
```
random_state = None
ridge = RidgeClassifierCV(alphas = np.logspace(-3, 3, 10))
sast_ridge = SAST(cand_length_list=candidate_lengths,
                          nb_inst_per_class=nb_inst_per_class,
                          random_state= random_state, classifier=ridge)
sast_ridge.fit(X_train, y_train)
print('score:', sast_ridge.score(X_test, y_test))
```

```
score: 0.9620991253644315
```

0.2.4 Using Shapelet Transform on Chinatown Dataset:

Similar to SAST, Shapelet transform uses a separate classifier to make the prediction of the class. This approach was applied to the Chinatown dataset with a RandomForestClassifier, and particularly the implementation of ShapeletTransform was imported directly from sktime library.

[ ]:
```
from sktime.transformations.panel.shapelet_transform import ShapeletTransform
```

Applying ST algorithm it was obtained a score of 0.97 in 1m40s approx. This difference in time, compared to SAST, it is due the fact that in SAST time series are chosen randomly for each class, in order to find the best subsequences that allows to make the classification.

[ ]:
```
#Creating a ShapeletTransform and fitting
st = ShapeletTransform(min_shapelet_length=min_shp_length,␣
 ↪max_shapelet_length=np.inf)
X_train_sktime = from_2d_array_to_nested(pd.DataFrame(X_train))
X_test_sktime = from_2d_array_to_nested(pd.DataFrame(X_test))
st.fit(X_train_sktime, y_train)
#Making the tranformation of time series and training a RandomForestClassifier␣
 ↪model
X_test_transformed = st.transform(X_test_sktime)
X_train_transformed = st.transform(X_train_sktime)
clf = RandomForestClassifier()
clf.fit(X_train_transformed, y_train)

print('Score:', clf.score(X_test_transformed, y_test))
```

```
Score: 0.9737609329446064
```

0.2.5 Using Catch22 on Chinatown Dataset:

Catch22 stands for Canonical Time-series Characteristics and it consist on transforming series into the 22 features, extracted from the hctsa toolbox. This approach is also available in sktime library so as to use it for the Chinatown dataset.

[ ]:
```
from sktime.classification.feature_based import Catch22Classifier
```

[ ]:
```
Catch22Classifier()
```

This approach got an score of 0.90 in accuracy and the process of creating Catch22Classifier instance and fitting the model took 32.6s approx.

[ ]:
```
catch = Catch22Classifier()
catch.fit(X_train,y_train)
print('score:', catch.score(X_test, y_test))
```

```
score: 0.9008746355685131
```

0.2.6 Using HIVE-COTE on Chinatown Dataset:

Lastly, HIVE-COTE is another time series classification algorithm that was applied to the Chinatown dataset. This approach is an ensemble of the STC, TSF, RISE and cBOSS classifiers which use different feature representations. The library used for this implementation was sktime in particular the hybrid classification methods.

[ ]:
```
from sktime.classification.hybrid import HIVECOTEV1
```

[ ]:
```
hive= HIVECOTEV1(n_jobs=-1)
hive.fit(X_train, y_train)
print('score:', hive.score(X_test, y_test))
```