SPC Module User Manual

Introduction

The Statistical Process Control (SPC) module is a python tool designed to monitor and control the quality of a process in real-time and detects anomalies or deviations from the expected behaviour of the process, allowing for timely intervention and maintenance of process quality.

This module is meant to be utilized jointly with a machine learning algorithm during the process of learning from data streams where concept drifts are expected.

Set Up

Download the *SPC_module* folder from https://github.com/miguelFigSilva/dsm_SPC.

Requirements

The user shall install pyhton 3 and the following packages:

- river
- matplotlib

Other commonly useful packages for using this module include:

- pandas
- sklearn

This can be done by running the following command:

pip install -r SPC_module/requirements.txt

Getting Started

To use this module, the user shall create a python script with an implementation like the one presented next.

Initialization

1. Import the necessary libraries. E.g.:

```
from SPC_module.SPC import SPCAlgorithm

import pandas as pd
import matplotlib.pyplot as plt
import os

import warnings

from river import compat
from river import metrics
from river import preprocessing
from sklearn import linear_model
from river import tree
```

2. Define the initialization functions for the intended estimator. These estimators should be compatible with the river library. Hint: there are available functions that help with this, e.g. *river.compat.convert_sklearn_to_river()*. E.g.:

```
def init_estimator_SGDClassifier():
    model = preprocessing.StandardScaler()
    model |= compat.convert_sklearn_to_river(
    estimator=linear_model.SGDClassifier(
        loss='log_loss',                    # 'log_loss' gives logistic regression
        eta0=0.01,
        learning_rate='constant'
    ),
    classes=[False, True]
    )
    return model
```

3. Load the data stream. Ensure that the dataset is in a compatible format. E.g.:

```
data_stream = pd.read_csv(f"{PATH}/data/synthetic_dataset.csv")
```

4. Initialize the SPCAlgorithm object with the chosen estimator. E.g.:

```
# The estimator should be river compatible! The user is responsible for ensuring this.
spc_detector = SPCAlgorithm(init_estimator_SGDClassifier)
```

Running the Algorithm

5. Once initialized, the SPC algorithm can run on the data stream. It will monitor the learning process of the chosen estimator and detect any anomalies or drifts. Run the algorithm. E.g.:

```
for i in range(data_stream.shape[0]):
    status, y, y_pred = spc_detector.model_control(data_stream, i)
```

Monitoring and Visualization

6. The SPC algorithm provides real-time feedback on the status of the process. It categorizes the process into three states:
    - Normal: The process is operating within expected parameters.
    - Warning: Deviation detected, indicating a potential issue.
    - Out of Control: Significant deviation detected, requiring intervention.

    The user may use the algorithm's output within the control loop. E.g.:

```
metric.update(y, y_pred)

if (i+1)%report == 0:
    print(f'{i+1} samples:', metric)

if status == 'Warning Level' and warn == -1 and i!=0:
    warn = i
    retrain = -1
    print(f'Warning after {i+1} samples')
elif status == 'Out-control' and retrain == -1 and i!=0:
    print(f'Re-train model after {i+1} samples')
    retrain = i
    warn = -1
else:
    warn = -1
    retrain = -1
```

7. The user may visualize the SPC algorithm results using the provided plotting functions. E.g.:

```
spc_detector.process_plot()
```

    This function should produce a figure with:
    - A plot for the error rate (including indicators for warning and drift levels).
    - A plot indicating the control state throughout the control process. E.g.: