anomalyDetectionEvaluation

August 1, 2024

[ ]:

```
from pathlib import Path
from timeeval import TimeEval,ResourceConstraints, DatasetManager,␣
 ↪DefaultMetrics, Algorithm, TrainingType, InputDimensionality
from timeeval.adapters import DockerAdapter
from timeeval.data_types import AlgorithmParameter
from timeeval.params import FixedParameters
from timeeval.resource_constraints import GB
from timeeval.params import ParameterConfig
import numpy as np
from timeeval.utils.window import ReverseWindowing
```

0.0.1 Load Datasets

[ ]:

```
custom_dataset = Path(r"data/datasets.json")
dm = DatasetManager(Path.cwd() / "data", create_if_missing=True,␣
 ↪custom_datasets_file=custom_dataset)
datasets = dm.select()
# datasets = dm.select(dataset='unsupervised_dataset.1')
```

0.0.2 Configure Algorithms post processing

[ ]:

```
# post-processing for LSTM-AD
def post_lstm_ad(scores: np.ndarray, args: dict) -> np.ndarray:
    window_size = args.get("hyper_params", {}).get("window_size", 30)
    prediction_window_size = args.get("hyper_params", {}).
 ↪get("prediction_window_size", 1)
    return ReverseWindowing(window_size=window_size + prediction_window_size).
 ↪fit_transform(scores)


# post-processing for EncDec-AD
def post_encdec_ad(scores: np.ndarray, args: dict) -> np.ndarray:
    window_size = args.get("hyper_params", {}).get("anomaly_window_size", 30)
    return ReverseWindowing(window_size=2 * window_size).fit_transform(scores)


# post-processing for DeepAnT
```

```
def _post_deepant(scores: np.ndarray, args: dict) -> np.ndarray:
    window_size = args.get("hyper_params", {}).get("window_size", 45)
    prediction_window_size = args.get("hyper_params", {}).
 ↪get("prediction_window_size", 1)
    size = window_size + prediction_window_size
    return ReverseWindowing(window_size=size).fit_transform(scores)
```

0.0.3 Configure Algorithms to Evaluate

[ ]:

```
algorithms = [
        Algorithm(
        name="LOF",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/lof",
            tag="0.3.1",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.UNSUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE
    ),
    Algorithm(
        name="KNN",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/knn",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.UNSUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE
    ),
    Algorithm(
        name="KMeans",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/kmeans",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.UNSUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE
    ),
    Algorithm(
        name="PCC",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/pcc",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.UNSUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE
    ),
     Algorithm(
        name="iForest",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/iforest",
            tag="0.3.1",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.UNSUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE
    ),
    Algorithm(
        name="IF-LOF",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/if_lof",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.UNSUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE
    ),
    Algorithm(
        name="LSTM-AD",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/lstm_ad",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.SEMI_SUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE,
        postprocess=post_lstm_ad
    ),
    Algorithm(
        name="Roburst PCA",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/robust_pca",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.SEMI_SUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE,
    ),
    Algorithm(
        name="FastMCD",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/fast_mcd",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.SEMI_SUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE,
    ),
    Algorithm(
        name="EncDec-AD",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/encdec_ad",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.SEMI_SUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE,
        postprocess=post_encdec_ad
    ),
    Algorithm(
        name="DeepAnT",
        main=DockerAdapter(
            image_name="ghcr.io/timeeval/deepant",
            tag="0.3.0",
            skip_pull=True
        ),
        data_as_file=True,
        training_type=TrainingType.SEMI_SUPERVISED,
        input_dimensionality=InputDimensionality.MULTIVARIATE,
        postprocess=_post_deepant
    ),
    ]
```

0.0.4 Configure evaluation run

set the number of repetitions of each algorithm-dataset combination (e.g. for runtime measurements):

set resource constraints

[ ]:

```
repetitions = 1000
rcs = ResourceConstraints(
    task_memory_limit = 3 * GB,
    task_cpu_limit = 1.0,
)
```

0.0.5 Configure TimeEval

[ ]:

```
timeeval = TimeEval(dm, datasets, algorithms,
                        repetitions=repetitions,
                        resource_constraints=rcs,
                        metrics=[DefaultMetrics.ROC_AUC, DefaultMetrics.PR_AUC,␣
 ↪DefaultMetrics.RANGE_PR_AUC],
                        force_dimensionality_match=True,
                        force_training_type_match=True
                        )
```

0.0.6 Run TimeEval and aggregate all Result

[ ]:

```
timeeval.run()
results = timeeval.get_results(aggregated=True)
results
```