CellPAD: Detecting Performance Anomalies in Cellular Networks via Regression Analysis

Introduction

How to accurately detect Key Performance Indicator (KPI) anomalies is a critical issue in cellular network management. We present CELLPAD, a unified performance anomaly detection framework for KPI time-series data. CELLPAD realizes simple statistical modeling and machine-learning-based regression for anomaly detection; in particular, it specifically takes into account seasonality and trend components as well as supports automated prediction model retraining based on prior detection results. We demonstrate how CELLPAD detects two types of anomalies of practical interest, namely sudden drops and correlation changes, based on a large-scale real-world KPI dataset collected from a metropolitan LTE network. We explore various prediction algorithms and feature selection strategies, and provide insights into how regression analysis can make automated and accurate KPI anomaly detection viable.

Publication

- Jun Wu, Patrick P. C. Lee, Qi Li, Lujia Pan, and Jianfeng Zhang.

  "CellPAD: Detecting Performance Anomalies in Cellular Networks via Regression Analysis"

  Proceedings of IFIP Networking, Zurich, Switzerland, May 2018.

(AR: 55/225 = 24.4%)[pdf]

People

This software is developed by Applied Distributed Systems Lab in the Department of Computer Science and Engineering at the Chinese University of Hong Kong (CUHK).

License

The source code of CellPAD is released under the GNU/GPL license.

Prerequsites

CellPAD is written in Python 3. To run CellPAD, please install the following packages first.

- Python 3.x;
- scikit-learn 0.19.0 or higher.

Installation

Run:

```
python setup.py install
```

Algorithm and Feature Selection

CellPAD integrates various statistical-based and machine-learning-based regression algorithms. They include:

- Statistical algorithms:
    - WMA, EWMA, and Holt-Winters (HW)

    - Local Correlation Score (LCS)
- Machine-learning-based regression regression:
    - Random Forest Regression (RF), Regression Tree (RT), Simple Linear Regression (SLR), and Huber Regression (HR).

How to call different algorithms?

```
DropController.detect(predictor)

ChangeController.detect(predictor)
```

- For **DropController**, "*predictor*" can be "RF", "RT", "SLR", "HR", "WMA", "EWMA", "HW".
- For **ChangeController**, "predictor" can be "RF", "RT", "SLR", "HR", "LCS".

How to perform feature selection?

```
DropController(feature_types=["Numerical"],
            feature_time_grain="Weekly",
            feature_operations=["Wma","Mean","Mean"])

ChangeController(feature_types=["Indexical"],
            feature_time_grain="Weekly",
            feature_operations=["Raw"])
```

- "*feature_types*" can be a subset of ["**Numerical**", "**Indexical**"]
- "*feature_time_grain*" can be a subset of ["**Hourly**", "**Daily**", "**Hourly**"]
- "*feature_operations*" can be a subset of ["**Raw**", "**Mean**", "**Median**", "**Wma**", "**Ewma**"]

How to remove any trend components?

```
DropController(to_remove_trend=True,
               trend_remove_method="center_mean")
```

```
ChangeController(to_remove_trend=False,
                 trend_remove_method="center_mean")
```

- "*to_remove_trend*" is **True** or **False** to indicate whether to remove the trend or not.
- "*trend_remove_method*" is "**center_mean**" or "**past_mean**".
    - "center_mean": the trend at time i is the mean of the points in [i-84,i+83].
    - "past_mean": the trend at time i is the mean of the points in [i-167,i].
    - Note that the accuracy of "past_mean" is slightly less than that of "center_mean" in general, as the latter considers the time interval closer to time i.

Examples and test data

We provide two examples on how to run CellPAD for anomaly detection of sudden drops and correlation changes. Run the following:

sudden drop

```
cd ./example
python example_drop.py
```

correlation change

```
cd ./example
python example_change.py
```