Opprentice: Towards Practical and Automatic Anomaly Detection Through Machine Learning

Authors:
Dapeng Liu†, Youjian Zhao†, Haowen Xu†, Yongqian Sun†, Dan Pei†, Jiao Luo‡, Xiaowei Jing§, Mei Feng§

†Tsinghua University, ‡Baidu, §PetroChina

Internet Measurement Conference (IMC ‘15)
Motivation

- Detecting performance anomalies is critical for Internet-based services
  - e.g., search engines, online shopping, and social networking
- Anomaly detection deployment problems
  - Dozens of anomaly detectors have been proposed
    - However, very little actual deployment
  - Portability problems
    - Parameters in one network don’t correspond to deployment in another
    - Time consuming/difficult task to “tune” internal parameters
      - End result = simple intuitive threshold with poor performance
Conflicting Challenges

- **Definition Challenge**
  - Difficult to precisely define anomalies in reality
    - Often impossible for operators to *quantitatively* define anomalies
    - Rather, operators prefer to describe anomalies *qualitatively*

- **Detector Challenge**
  - Reasonable detection accuracy requires both algorithm expertise and domain knowledge
    - Time consuming to tune parameters and thresholds
Proposed Solution

- Opprentice (Operators’ apprentice)
  - Operators only manual work is to periodically label the anomalies in the performance data
    - Authors design a convenient labeling tool
    - Very visual rather than a series of questions
    - Assumption that anomalies are less frequent than normal activity
  - Operators can input simple desired accuracy preference
    - e.g., recall and precision greater than 0.66
    - Opprentice automatically configures itself to satisfy such desires
  - Applies basic detectors as the features of the detector
    - Features and labels are used to train a Random Forest classifier
Proposed Solution (cont.)
Input Data

- Detector works with any time series data
  - (time, value) pairs
  - i.e., Key Performance Indicator (KPI) data
  - e.g., SNMP, syslogs, network traces, web access logs, etc
- Authors study three “representative” KPIs for a search engine
  - Search Page View (PV)
  - Number of slow responses of search data centers (#SR)
  - The 80th percentile of search response time (SRT)
    - Representative in that they vary in seasonality and dispersions
Input Data (cont.)

- **Seasonality**
  - e.g., PV is much more regular than the other two

- **Dispersions**
  - Coefficient of variation
    - i.e., standard deviation divided by the mean

- **Different KPIs have different anomalies**

```
<table>
<thead>
<tr>
<th></th>
<th>PV</th>
<th>#SR</th>
<th>SRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval (minute)</td>
<td>1</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>Length (week)</td>
<td>25</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>Seasonality</td>
<td>Strong</td>
<td>Weak</td>
<td>Moderate</td>
</tr>
<tr>
<td>$C_v$</td>
<td>0.48</td>
<td>2.1</td>
<td>0.07</td>
</tr>
</tbody>
</table>
```
Opprentice Goals

- **Recall**
  - # of true anomalous points detected / # of true anomalous points

- **Precision**
  - # of true anomalous points detected / # of anomalous points detected

- **Quantitative Goal**
  - Achieve operators accuracy preference
    - e.g., Recall & Precision > 0.66

- **Qualitative Goal**
  - Automatic
    - Operators are not involved in selecting or tuning detectors
Machine Learning Challenges

- **Labeling Overhead**
  - Oppreintice only takes 6 min/month of training data to label

- **Incomplete Anomaly Cases**
  - Oppreintice incrementally retrains the classifier with new data

- **Class Imbalance**
  - Oppreintice automatically adjusts classification threshold (cThld)

- **Irrelevant and Redundant Features**
  - Oppreintice uses Random Forests as an ensemble learning algorithm
Design Details

(a) Training classifier.

(b) Detecting anomaly.
Labeling Tool

- Displays KPI data as a line graph
  - Data of the last day and last week is also shown to compare
- Operators do not have to label each time bin one by one
  - But still can label more than identified by detectors
Basic Detectors as Feature Extractors

- Represent different detectors with:
  
  data point \xrightarrow{\text{a detector with parameters}} \text{severity} \xrightarrow{sThld} \{1, 0\}

  - Severity measures how anomalous the data point is
  - sThld is the threshold for classification

- Detectors have a set of internal parameters
  - e.g., Exponentially Weighted Moving Average (EWMA) uses alpha to represent how important recent data is

- Opprentice takes 14 such statistical detectors and “sweeps” their various configurations to create 133

- Each sThld acts as a feature of the data point
## Basic Detectors as Feature Extractors (cont.)

<table>
<thead>
<tr>
<th>Detector / # of configurations</th>
<th>Sampled parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple threshold [24] / 1</td>
<td>none</td>
</tr>
<tr>
<td>Diff / 3</td>
<td>last-slot, last-day, last-week</td>
</tr>
<tr>
<td>Simple MA [4] / 5</td>
<td>win = 10, 20, 30, 40, 50 points</td>
</tr>
<tr>
<td>Weighted MA [11] / 5</td>
<td>$\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$</td>
</tr>
<tr>
<td>MA of diff / 5</td>
<td></td>
</tr>
<tr>
<td>TSD [1] / 5</td>
<td></td>
</tr>
<tr>
<td>TSD MAD / 5</td>
<td>$\text{win} = 1, 2, 3, 4, 5 \text{ week(s)}$</td>
</tr>
<tr>
<td>Historical average [5] / 5</td>
<td></td>
</tr>
<tr>
<td>Historical MAD / 5</td>
<td></td>
</tr>
<tr>
<td>Holt-Winters [6] / $4^3 = 64$</td>
<td>$\alpha, \beta, \gamma = 0.2, 0.4, 0.6, 0.8$</td>
</tr>
<tr>
<td>SVD [7] / $5 \times 3 = 15$</td>
<td>$\text{row} = 10, 20, 30, 40, 50 \text{ points, column} = 3, 5, 7$</td>
</tr>
<tr>
<td>Wavelet [12] / $3 \times 3 = 9$</td>
<td>$\text{win} = 3, 5, 7 \text{ days, freq} = \text{low, mid, high}$</td>
</tr>
<tr>
<td>ARIMA [10] / 1</td>
<td>Estimation from data</td>
</tr>
</tbody>
</table>

In total: 14 basic detectors / 133 configurations
Random Forest as Classifier

- Need to be careful in selecting a ML algorithm
  - Many algorithms perform poorly with redundant and irrelevant features
    - e.g., Naïve Bayes, logistic regression, decision trees, linear SVM, etc
- Random Forests
  - Ensemble method that has been shown to be robust to this
  - Built out of many “randomized” decision trees
    - Each tree is trained on a subset of the training data
    - Trees only consider a random subset of the features each time
  - Fix overfitting of decision trees
  - Only input is the classification threshold (cThld)
Configuring cThld

- **Precision-Recall Curves (PR)**
  - Show the trade off between Precision and Recall over various choices for cThld

- **F-Score**
  - \( \frac{2 \times p \times r}{p + r} \)
  - Used to find an optimal cThld
  - SD(1,1) is another such metric

- **Authors introduce new metric**
  - Preference Centric Score (PC)
  - \( \text{PC}(r,p) = \text{FS} + 1 \), if \( r > R \) and \( p > P \)
  - FS, otherwise
Configuring cThld (cont.)

- **PC-Score requires “oracle” mode**
  - Post-priori

- **K-fold Cross-validation**
  - Split training data into parts to find “best” cThld value
  - However, the best cThld can greatly differ over weeks
    - Past indications do not necessarily tell the best for current

- **EWMA Based cThld Prediction**
  - Update cThld each week but utilize recent predictions

\[
c_{Thld}^{p}_{i} = \begin{cases} 
\alpha \cdot c_{Thld}^{b}_{i-1} + (1 - \alpha) \cdot c_{Thld}^{p}_{i-1} & , i > 1 \\
5\text{-fold prediction} & , i = 1 
\end{cases}
\]
Configuring cThld (cont.)
Evaluation

- **Accuracy of Random Forest**
  - Against basic detectors and static combinations of basic detectors
  - Against other ML algorithms

- **Incremental Retraining**
  - Want to show that continually updating classifier is needed

- **PC-Score vs. Other Accuracy Metrics**
  - Want to show that the author’s configuration of cThld better predicts the accuracy preferred by the operator

- **EWMA vs. 5-Fold for cThld Prediction**
  - Want to show that cThld should be updated based on recent settings

- **Labeling time vs. Tuning Time**
  - Want to show that Opprentice meets its qualitative goal
Accuracy of Random Forest

(a) KPI: PV. The basic detector ranking first in AUCPR is TSD MAD (win = 5 weeks), the 2nd one is historical MAD (win = 3 weeks), and the third one is TSD (win = 5 weeks).

(b) KPI: #SR. The basic detector ranking first in AUCPR is simple threshold, the 2nd one is SVD (row = 50, column = 3), and the third one is wavelet (win=3, freq=low).

(c) KPI: SRT. The basic detector ranking first in AUCPR is SVD (row = 20, column = 7), the 2nd one is TSD MAD (win = 3 weeks), and the third one is TSD (win = 2 weeks).
Accuracy of Random Forest (cont.)

(a) KPI: PV  Number of features used for training

(b) KPI: #SR  Number of features used for training

(c) KPI: SRT  Number of features used for training
Incremental Retraining

(a) KPI: PV
(b) KPI: #SR
(c) KPI: SRT

ID of 4-week moving test sets
PC-Score vs. Other Accuracy Metrics

(a) KPI: PV
PC-Score vs. Other Accuracy Metrics (cont.)

(b) KPI: #SR
PC-Score vs. Other Accuracy Metrics (cont.)

(c) KPI: SRT
EWMA vs. 5-Fold for cThld Prediction

(a) KPI: PV
EWMA vs. 5-Fold for cThld Prediction (cont.)

(b) KPI: #SR
EWMA vs. 5-Fold for cThld Prediction (cont.)

(c) KPI: SRT
Labeling Time vs. Tuning Time

![Graph showing the relationship between labeling time (in minutes) for one-month data and the number of anomalous windows in one-month data. The graph includes data points for different categories labeled as PV, #SR, and SRT.](image)