Anomaly Detection and Noise Filtering in Time Series Data

Alinur Caglayan

Dec 15th, 2019

This report is written to explain my solution to offsite task for Algorithm Engineer position in Pubinno. The task was challenging and fun, it gave me the opportunity to explore data science and digital signal processing areas which I've never worked on before.

Problem

When I first read the task, as an electrical and electronics engineer initially I thought about adding an additional hardware filter. But adding a hardware filter would also kill the chance to see possible anomalies, since it will filter out all the signals up or below a certain threshold. Then I've widened my research and saw that it was about data science, which I have little to no clue about. The topics I've focused on was anomaly detection and signal smoothing.

Techniques

It's a widely used technique to identify trends without disturbing the signal's pattern a lot, which I need in this task. There are 2 commonly used filters to smooth the signal,

- **Moving Average Filter:** Smooths the data by replacing each data point with the average of its neighbors within a window size. Good at dealing with noise but may introduce transient effects and effect trends.
- **Savitzky-Golay Filter:** Tracks the signal closely and accounts for transient effects, preserves high frequency components of data while smoothing the signal.

Because my lack of practical idea about these techniques, I've looked for ways to get my hands dirty in Python. I've found implementations of these filters in SciPY and Pandas libraries. Let's dig into them,

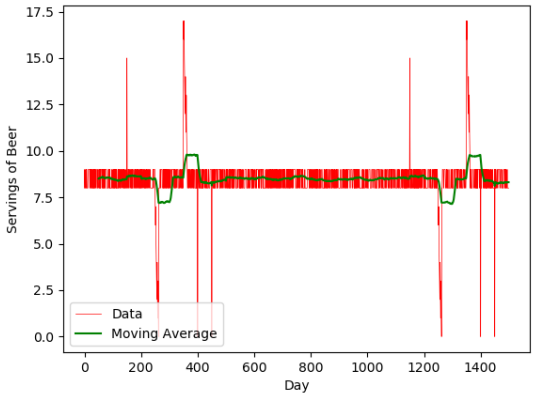

Figure 1 – Moving Average Filter, WS = 51

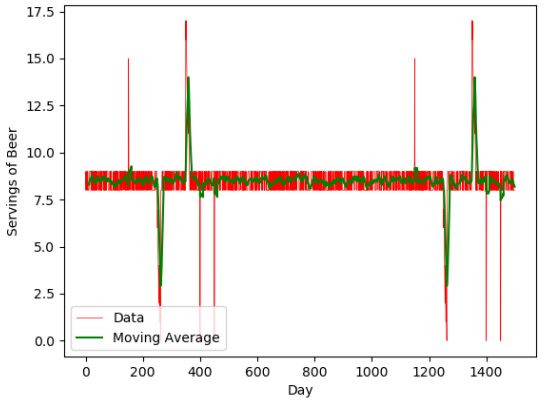

Figure 2 – Moving Average Filter Window Size = 11

Figures 1 and 2 are the output graphs of implementation of moving average filter with a window size of 51 and 11 for 1500 samples of data. While it deals with the noise well, there is a huge problem with this technique. If I decrease the window size, it gets bad in dealing with noise; I increase the window size, it gets bad at catching the trends on time. Besides, it changes the data values, while this gives us a clue about what's going on, it takes our chance to access the raw values. We need more space to store the raw values if needed, which is not a good practice if we have a limited resourced embedded system.

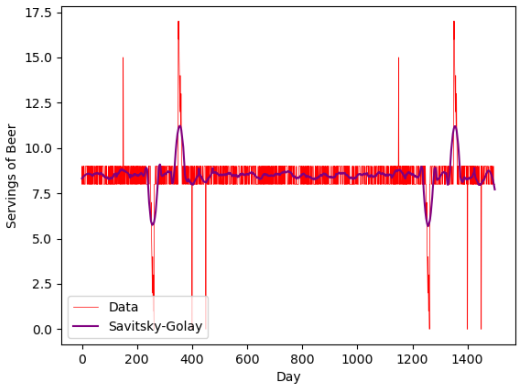

Figure 3 – Savitzky-Golay Filter, WS = 51

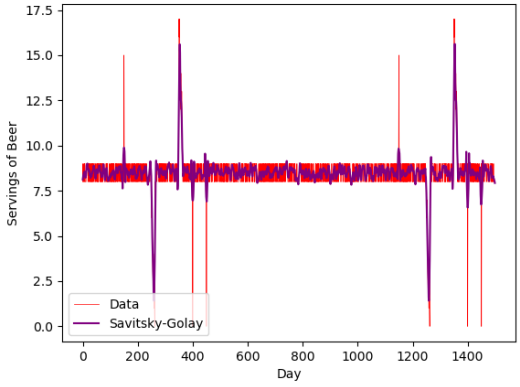

Figure 4 – Savitzky-Golay Filter Window Size = 11

Figures 3 and 4 are the output graphs of implementation of Savitzky-Golay filter with a window size of 51 and 11 for 1500 samples of data. As opposite of moving average filter, Saviztky-Golay filter doesn't deal with the noise as well as moving average filter, but it's definitely better at catching the trends on time. Besides, it changes the data values, while this gives us a clue about what's going on, it takes our chance to access the raw values. We need more space to store the raw values if needed, which is not a good practice if we have a limited resourced embedded system.

So I tried to make a filter, which has the same principle as moving average filter, but only eliminates noise and minimize the modification of anomaly data.

```
def deviated(value, mean, threshold):
  return abs(safe_divide(max(value, mean), min(value, mean))) > threshold

def filter(x, window, threshold):
  for i in range(len(x)):
    mean = window_mean(x, i, window)
    if deviated(x[i], mean, threshold):
      x[i] = mean
```

Figure 5 – Code of my first filter

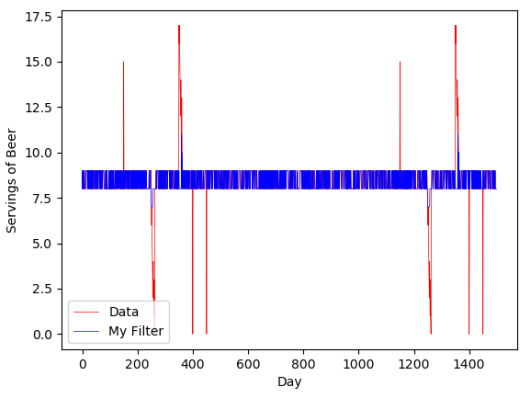

Figure 6 – My Filter, WS = 51

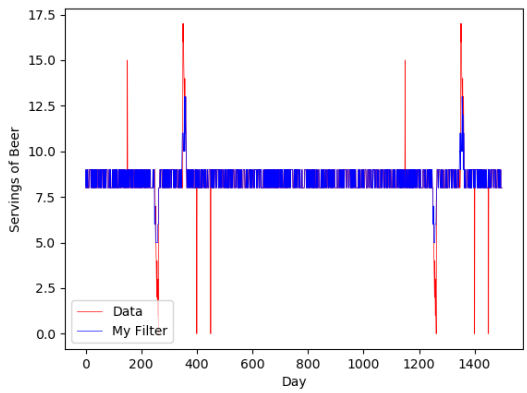

Figure 7 – My Filter, WS = 11

As we can see, my filter deals with the noise, but minimizes the modification to abnormal trends. It gets better when window size is decreased, which is opposite of both pre-implemented Moving Average and Savitzky-Golay filters. Because it doesn't touch the datum if there's the deviation from mean value is below the threshold, which is 1.3 in this case.

So for my filter seems fine. It first goes through data and filters the spikes.

```
# filter method continued
# Find the trend and toggle the LED
  global_mean = get_mean(x)
  trend_window = 2
  led_indicator = False

  for i in range(len(x)):
    direction = trend_direction(x, i, trend_window, global_mean, threshold)
    if direction > 0 and led_indicator is False:
      led_indicator = True
      print("TURN ON LED", i)
    elif direction < 0 and led_indicator is True:
      led_indicator = False
      print("TURN OFF LED", i)
  return x

def trend_direction(x, idx, window, global_mean, threshold):
  num_increase = 0
  num_decrease = 0
  for j in range(idx + 1, min(idx + window, len(x))):
    if deviated(x[j], global_mean, threshold):
      if x[j] > global_mean:
        num_increase += 1
      else:
        num_decrease += 1
  if num_increase >= window - 1:
    return 1
  elif num_decrease >= window - 1:
    return -1
  return 0
```

Figure 8 – Finding trends and toggling the LED

Than it comes to finding the anomalies, as I consider as sales trends. My filter goes through data again, and since the data is noise-free now, all ups and downs can be considered as anomalies. By finding the direction of trend, I toggled the LED on or off. I had a better solution though, having 2 LED's for both directions is a better indicator since we toggle only on increase in sales, yet decrease in sales is a worse situation and should be handled more quickly. I don't toggle any LED if the sales are in normal range.

Criticism

Although my filter seems fine, it has a huge flow, it may not be compatible with different cases, because I use a threshold to determine the deviation. The main compatibility problem comes from the variance. In my data set, there are lots of eights and nines, spikes are 15 and 0 and anomalies are visibly high or low. The problem would rise if I had a high variance dataset. If I had zeroes and tens as data, all data would be equaled to mean value of 5, since all would be deviated more than threshold, even if it was a normal situation for that place.

I've thought and read a lot on this threshold problem, and my conclusion is that while a constant threshold, like >10, <2, >5 is bad, because every bar has different customer mass and what is considered low sales trend for a bar, can be considered as high sales trend for another one. Yet I think I always have to put a relative threshold, like >1.3x, <0.7x, at least at the beginning. Because initially, we don't have any idea on what shall be considered as anomaly for that place. After we get a huge data set and have an idea on what we can consider as anomaly, what we can consider as in normal interval.

It was great to research on this subject, thank you for giving me the opportunity.