COMP3217 Security of Cyber Physical Systems

Option A: Detection of Manipulated Pricing in Smart Energy CPS Scheduling

Thomas David Hoad

May 2022

Contents

1 Introduction

This report details the data classification and energy scheduling methods I used for solving Option A for the detection of manipulated pricing in smart energy CPS scheduling. For this, I decided on using the K-nearest neighbours algorithm paired with a two-stage simplex algorithm, both coded in Java.

2 Data Classification Algorithm

The classification of the data can be broken down into the main stages. Training the classification algorithm and then using it to classify the training data.

Training the Classification

The main portion of training the classification was to find the mean normal curve, the mean abnormal curve and the max and minimum values for each curve value. The mean normal curves and the mean abnormal curves are used to give each training data curve a weighting that's value is equal to the euclidean distance from it to either the mean normal or mean abnormal curve, depending on what its set value is. The purpose of this is to give a higher value to those points that are the most normal and less value to those that are closer to the decision boundary of normal or abnormal. The euclidean distance for calculating weighting uses normalised values.

$$w = \sqrt{(x'_n - x'_1)^2 + (x'_n - x'_2)^2 + ... + (x'_n - x'_{24})^2}$$

This part of the program collects the maximum and minimum values for the training curves given. These values are very important as we use it to feature scale each data point to a value within the range [0, 1].

$$x' = \frac{x - min}{max - min}$$

Min-max feature scaling is very important for k-nearest neighbours to normalise our data to reduce the bias of certain variables. This also means that our values are all within a very distinct range than can be controlled easier.

K-Nearest Neighbours

As mentioned, the classification algorithm I have used for my solution is the k-nearest neighbours algorithm. I decided on using this algorithm because it is a supervised learning algorithm, can be used for multi-dimensional data and has many ways of being improved to increase the classification accuracy. For this solution to be effective I had to find a reasonable value for K that didn't create over-fitting or under-fitting. I tested values within the range of 0 to 100 and came to the conclusion that the best value was 50. This gave me the highest accuracy when testing the algorithm on the training dataset.

K-nearest neighbours works by finding the K nearest points to the current point and determining if the majority of them are normal or abnormal. In this scenario, I have turned every curve into a 24 dimensional vector and use the euclidean distance between these points to determine what points are close together.

Bagging was an important part of this solution to improve the accuracy of my solution. I partitioned the training dataset into 20 and used these partitions to perform k-NN several times to collect 10 values for each testing pricing curve. I could then aggregate the values to get an average result for the curve. This would reduce the effect of outliers in the training dataset. This was particularly helpful as it minimises the effects of having a less than optimal value for K.

As spoken previously, I use min-max feature scaling to cluster the data points evenly and normalise the scales for each dimension. I have also used weighting to higher value those curves closer to their normal or abnormal mean.

$$bagValue = \begin{cases} normal, & normalCount > \dfrac{k-1}{2} \\ abnormal, & otherwise \end{cases}$$

$$curveValue = \begin{cases} normal, & bagNormalCount > \dfrac{b-1}{2} \\ abnormal, & otherwise \end{cases}$$

Classifying the Training and Testing Data-sets

To determine the accuracy of my algorithm, I tested the algorithm first on the training dataset. From this solution, I got an accuracy of 81.89% which seems quite reasonable.

Classifying the Testing Results

When running the algorithm on the testing dataset, I got 50 normal and 50 abnormal pricing curves. Shown below is the list of all abnormal programs. After the classification, a bar chart is made for every program. An example can be found in appendix A and the rest have all been included within the source code.

- LP0
- LP1
- LP2
- LP4
- LP5
- LP8
- LP11
- LP14
- LP16
- LP17
- LP19
- LP20
- LP23
- LP25
- LP28
- LP29
- LP31
- LP32
- LP34
- LP35
- LP37
- LP39
- LP41
- LP44
- LP45
- LP46
- LP47
- LP51
- LP52
- LP58
- LP60
- LP61
- LP65
- LP68
- LP73
- LP74
- LP76
- LP77
- LP78
- LP80
- LP81
- LP85
- LP87
- LP89
- LP90
- LP92
- LP94
- LP95
- LP97
- LP98

3 Linear Programming Algorithm

To perform the linear scheduling of my abnormal pricing curves, I decided to use the two-stage simplex algorithm to obtain the results. This involved creating a set of constraints using the input spreadsheet and using the pricing curve for the minimisation function.

Simplex Constraints

Shown below is an example for each of the type of constraints for program 0 which was classified as an abnormal program. The first equation shows the start of the minimisation function, the second expression shows one of the scheduling equality constraints and the third shows the variable bounds constraint. There are total of 50 scheduling constraints calculated from the input spreadsheet and about 350 variable bounds constraint.

$$c = (4.51285340120281 * user3\_task10\_time0) + ...;$$

$$user4\_task7\_time22 + user4\_task7\_time23 = 1;$$

$$0 <= user1\_task1\_time20 <= 1;$$

For the simplex algorithm, I needed to convert these equations into those that could be minimised in a simplex tableau. The variable bounds each get converted into an equality with a slack variable.

$$user1\_task1\_time20 + s1 = 1;$$

The scheduling constraints require a surplus variable and an artificial variable for each constraint because they are equality constraints.

$$user4\_task7\_time22 + user4\_task7\_time23 - s2 + a1 = 1;$$

Before we can create the tableau, we also need a function to minimise the artificial variables. Given the presence of artificial variables, we will need to perform two-stage simplex.

$$I = -(a_1 + a_2 + ...a_n)$$

We will also need to rewrite our original minimisation function as a maximisation function to account for this to give us the total set of constraints that form our initial tableau.

$$C = -D$$

$$D = (4.51285340120281 * user3\_task10\_time0) + ... + s_1 + ... + s_n - I;$$
$$I = -(a_1 + a_2 + ...a_n)$$

Simplex Results

The tableau is then created as a 2 dimensional array and the simplex algorithm I have implemented into my program runs on it. The results of the algorithm are written to a file called results.txt. The standard results I got were about 350 for each program after the second pass. When checking my results using LP-Solve, I should have been getting about 450. Shown in appendix B is the energy scheduling results I got for each of the abnormal programs after the first and second stages of simplex. Each of the results has been brought to 2d.p. for simplicity.

4 Conclusions

Overall, I think the strongest part of my solution is the classification algorithm. An accuracy of 0.8189 is fairly confident and with a 50 50 split from the testing data, this is what I would expect. The energy scheduling solution seems to be giving me consistently similar results, but not the exact results I would expect. This could be an issue with creating the tableau or the method of simplex algorithm I am using, but my results are at least within the correct scale and region.

This solution could have been improved by using more accurate machine learning classifications and using prebuilt libraries to reduce the work required for energy scheduling. There are so many ways of classifying the data, but I am satisfied with the results I achieved.

A Example Abnormal Chart

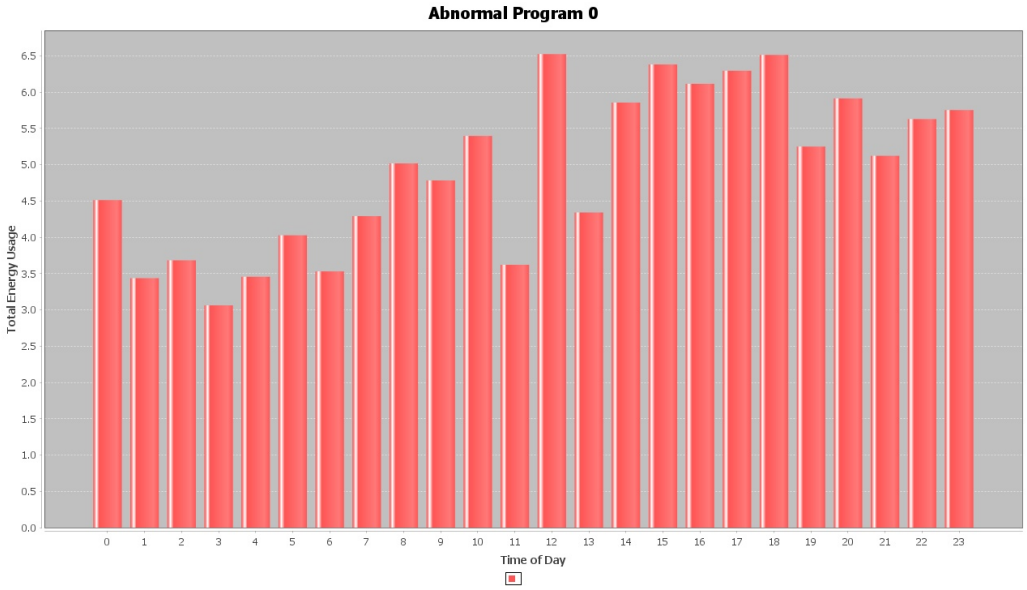

Figure 1: Example Classified Program

B Scheduled Abnormal Programs

- LP0 — First Stage: 492.91 — Second Stage: 354.045
- LP1 — First Stage: 491.42 — Second Stage: 356.48
- LP2 — First Stage: 487.35 — Second Stage: 354.96
- LP4 — First Stage: 490.88 — Second Stage: 355.75
- LP5 — First Stage: 484.56 — Second Stage: 354.59
- LP8 — First Stage: 487.52 — Second Stage: 352.70
- LP11 — First Stage: 484.02 — Second Stage: 352.68
- LP14 — First Stage: 496.34 — Second Stage: 357.99
- LP16 — First Stage: 489.18 — Second Stage: 352.72

- LP17 — First Stage: 490.43 — Second Stage: 352.86
- LP19 — First Stage: 489.10 — Second Stage: 354.23
- LP20 — First Stage: 493.71 — Second Stage: 352.46
- LP23 — First Stage: 495.12 — Second Stage: 358.69
- LP25 — First Stage: 484.88 — Second Stage: 352.61
- LP28 — First Stage: 492.22 — Second Stage: 358.36
- LP29 — First Stage: 490.79 — Second Stage: 358.00
- LP31 — First Stage: 493.52 — Second Stage: 356.03
- LP32 — First Stage: 491.71 — Second Stage: 358.09
- LP34 — First Stage: 489.12 — Second Stage: 356.64
- LP35 — First Stage: 489.07 — Second Stage: 356.29
- LP37 — First Stage: 483.38 — Second Stage: 350.84
- LP39 — First Stage: 499.11 — Second Stage: 359.74
- LP41 — First Stage: 484.84 — Second Stage: 352.40
- LP44 — First Stage: 485.10 — Second Stage: 352.90
- LP45 — First Stage: 500.47 — Second Stage: 361.36
- LP46 — First Stage: 489.83 — Second Stage: 357.31
- LP47 — First Stage: 489.39 — Second Stage: 352.71
- LP51 — First Stage: 485.57 — Second Stage: 355.69
- LP52 — First Stage: 495.75 — Second Stage: 356.57
- LP58 — First Stage: 497.77 — Second Stage: 356.18
- LP60 — First Stage: 487.96 — Second Stage: 354.54
- LP61 — First Stage: 488.78 — Second Stage: 356.56

- LP65 — First Stage: 485.75 — Second Stage: 353.29
- LP68 — First Stage: 490.21 — Second Stage: 355.00
- LP73 — First Stage: 490.47 — Second Stage: 355.62
- LP74 — First Stage: 487.35 — Second Stage: 356.28
- LP76 — First Stage: 495.31 — Second Stage: 357.64
- LP77 — First Stage: 489.31 — Second Stage: 353.89
- LP78 — First Stage: 483.27 — Second Stage: 352.72
- LP80 — First Stage: 488.25 — Second Stage: 357.88
- LP81 — First Stage: 482.48 — Second Stage: 352.91
- LP85 — First Stage: 484.55 — Second Stage: 355.19
- LP87 — First Stage: 490.25 — Second Stage: 356.05
- LP89 — First Stage: 492.29 — Second Stage: 356.30
- LP90 — First Stage: 483.70 — Second Stage: 353.19
- LP92 — First Stage: 488.04 — Second Stage: 353.01
- LP94 — First Stage: 488.58 — Second Stage: 354.50
- LP95 — First Stage: 490.53 — Second Stage: 352.57
- LP97 — First Stage: 486.94 — Second Stage: 354.41
- LP98 — First Stage: 493.78 — Second Stage: 353.25