Project Seminar
Flood Prediction at German corner in Koblenz

Long Phan [1],

Supervisor: Prof. Dr. Thomas Goetz [1]

[1]Department of Mathematics, University of Koblenz, Germany
*E-mail: ldphan@uni-koblenz.de

Abstract

Flooding events have been happening occasionally especially when heavily raining or affection of bad weather condition. The aim of this project seminar is to understand history data of water level measured during the flood periods and concurrently develop methods and tool to support the decision-maker.

1 Introduction

Flooding has been always being the known natural disaster around the world during history until then. Events in the year 1993, 1995, 2011 under report confirmed the flood at city Koblenz. In our use case, the location is informed at German corner and in extreme weather condition the decision-maker is required to give a fast decision (e.g. call the crane to remove the container and its facilities) at the right time. A decision too early or late would cause a lot of costs and damages overall. To achieve our goal, firstly we proceed step by step by analysing issue and understanding the given data delivered by the city, then choosing the appropriate meta data to derive the necessary information and by using these information we're able to develop the approaches to predict the water level at expected time in hours of day.

Figure 1: High water in Koblenz 1993.

2 Flood issue and data understanding

Following the given data, the position of container on the map is located on the land with height 67 meters to sea surface level and **PNP's**[1] position around this location is determined with 57 meters. So the distance between the PNP and the container is about 10 meters (67 - 57). So the safe distance must be at least $<$ 10 meters to keep water level away from position of container. Based on the given data of the measuring water level, we confirmed that many times in history the location of container has been flooded when water level was rising greater than 10 meters (see: figure 2). The first question comes up that how do we know at what water level should we start concerning. Therefore, the first algorithm has been developed to classify the water level into 3 controlling zones: **"Green (safe), Yellow (warning) and Red (alarm)"** [6]. After we have classified the water level, we will able to identify whether or not the water level at the current time need to be concerned and the predicting calculate water level might be in danger zone or still in safe range.

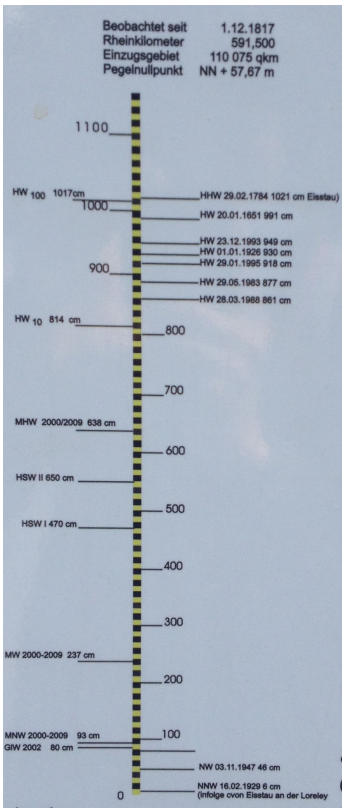

Figure 2: Pegel Koblenz Rhein.

There are 3 important parameters which will be used during the whole work:

The current water level, the current time point and the predict hours. These 3 parameters will be input by user to keep an connection between history data and the reality which is happening outdoor. Next we will step by step apply this first achieved information into the methods to find out the final answer for the question: at what water level should the decision-maker call the crane to remove the container ?

[1] https://de.wikipedia.org/wiki/Pegel

3 Approaches

Method 1

The idea is to classify data to identify at what water level would be considered as *dangerous, warning or safe* by calculating the distance of every point to the major pre-defined points (MAX_waterlevel, MIN_waterlevel, Average_waterlevel). E.g. if the distance of water point to Min_waterlevel is smallest comparing to other points (MAX_waterlevel and Average_waterlevel), this water point will belong to the Green zone and so on. By applying this way to all measured water point, we then know which group every water point belongs and the range of every water zone (Red, Yellow, Green) and finally achieve the first conclusion that at what water level we should start concerning. Result will be 3 lists illustrating 3 groups (Red, Yellow, Green). The visualization programming technique will show us clearly the distinguishing zones.

Algorithm 1 Data Classification

```
1:  preprocessing step by converting real data into normalized data using method min-max with
    pre-defined boundary [0, 1] [5] {see: functions normalizedata and _convertrealnorm}

2:  dist_red = dist_yellow = dist_green = 0 {initiate distance = 0}
3:  for elem in allwaterlevelpoints do
4:      dist_red = MAX_waterlevel - elem_value
5:      dist_yellow = Average_waterlevel - elem_value
6:      dist_green = MIN_waterlevel - elem_value

7:      if dist_red < dist_yellow and dist_red < dist_green then
8:          elem belongs to red_group {red_group is alarm group}
9:      else if dist_yellow < dist_red and dist_yellow < dist_green then
10:         elem belongs to yellow_group {yellow_group is warning group}
11:     else if dist_green < dist_yellow and dist_green < dist_red then
12:         elem belongs to green_group {green_group is safe group}
13:     else
14:         elem cannot be distributed
15:     end if
16: end for
```

Computation Time: is proceeded mainly from calling for-loop on all water point (see: line 3), so it's limited by upper bound of polynomial-time.

Next idea is to find and calculate the value of all points at every single time points (in time step 15 minutes) from 0:00 to 24:00, so the total calculated time points is 96. It's expected that at the same time (o'clock) but different time slots (days, months, years) the water level points have different values, these are result measured in history. So we will need a way to reuse all of these data and avoid losing information as much as possible and concurrently keep staying closely with the reality. In this case, the current water level which is representing the reality and total water data points (measured with current data set is 17043 points) are the major parameters. All the data points need to be reordered and prioritized with respect to this current water level. The water point which is closest to the current water level will contain the most valuable information, so it has the highest priority during computing. Furthermore, the water points before and after the current time point and least recently to the current time points will be considered with the most valuable information (e.g. time 15 minutes before and after the current time point will get the more important information rather than 30 minutes before and after). After that we can identify the already calculated data points and apply least square method on these dataset to derive the corresponding line which is closest to the data points. By using the built-in function *polyfit* of Numpy-Python framework, the implementation demand becomes easier and the same task can be repeated with the same method but on different degree so that 2 different results (one as linear result and one as parabolic result) will be compared each other. The error estimate method is based on recalculation of the current water level to find out the equation form which has smallest error estimate will be chosen. Finally, we're able to calculate the predicting water level from the computed equation by inputting the expected hours at later time. The difference between the calculated water level and current water level will be reused to create the artificial predicting point. This is the idea to keep the reality and history data as close as possible. The following second algorithm describes again how to put all above ideas together to calculate the predicting water level.

Computation Time: is proceeded mainly from calling for-loop on all water point in all time points, so it's limited by upper bound of polynomial-time.

(e.g. $NumberOfDays181days \times NumberOfElem17043points \times NumberOfTimeSlots96timepoints$)

Algorithm 2 Prediction by applying method least square data fitting
For full details on implementation, see: function lspm and _task_lspm

{*return will be a list of 96 points. Every point contains a list of all measured water level points. see: function _findWaterlevel*}
for $data$ in $dataset$ **do**
    Group by and sort all data by time from 0:00 to 24:00,
end for
Sort this list with respect to current water level by calculating distance between the water point and current water level. The smallest is the distance, the higher is the rank of the water point. Return will be a sorted list {*See: function _findAllPoints*}
{*the number of time step is 96, with step size 15 minutes from 0.00 to 24.00. See: function calMeanValue_prio)*}
for idx in timesteps **do**
    sublist = find length (number of points) at every single time point and calculate sum from 1 to this length
    sum = sum + sublist
    for $elem$ in $dataset$ **do**
        result = (lengthsublist - i)*value_thispoint/(sumsublist)
        {*calculate sum of coefficients water points at every time point by using this. The result will be returned as a list of all sub results*}
        final_result.append(result)
        {*save result at every time point into a list*}
    end for
end for
{*result will be a list of all calculated water point at 96 time points*}
Identify the time range where the calculated water points are being found with respect to current time point. A sub list will be returned based on the predicting hour forward and backward. {*e.g. predicting hour is 8 hours and current time is 10:00, so the calculated water points will be gathered from 2:00 to 18:00. The returning result is a list of data points which will be used for finding equation in next step (see: function _processZone_prio)*}
Apply method least square data fitting to above list to find the coefficients, respectively equation which is most closest to the data points, choose the best result with least error estimate [4]
Calculate the water level with the equation by giving the predicting hours
{*see: function _task_lspm)*}
Recalculate water level at current time
Find the error difference between the computed water level and the inputting water level at the current time
Find artificial result by applying this error difference at the predicted time
{*see: function showResult)*}
Visualize all results
{*see: function visualize*}

Method 2

The further idea is to sort out all given dataset in years *1993, 1995, 2013, 2016* in a meaning order, so that the rate of changes of water level by days will be recognized clearly. The first result we got is the total number of days *181 days* and every day is considered as a store-object which contains the data about water level was measured in every 15 minutes, should be from 0:00 to 23:45. Data will be tracked by days and timing which is suitable to find out the relationship between water level at the current and history time. The next algorithm helps to find all days with water levels which are close possibly to the current water level at the current time by rounding up the normalized values of water levels and comparing these values each other of water level. Therefore we can identify the worst case and best case when the water level keep rising or decreasing depending on the current weather condition outside. After all, we apply the same technique as in first method to find out mean value, coefficient and calculate the predicting water level and create the artificial water level in expected predict hours.

Algorithm 3 Find out all data sorted by days and time
For full details on implementation, see: function rateofchange)

1: **for** $data$ in $dataset$ **do**
2: group by dataset sorted by date (expected data from 0:00 to 24:00)
3: **end for**
4: return a sorted list by days of rate of changes of water level

Algorithm 4 Prediction by finding out the data which are closest with the current time and water level. *For full details on implementation, see: function _findcritiquepoint*

1: **for** $data$ in $sortedlistfromabovealgorithm$ **do**
2: **if** the time at day == the current time and the rounded up water level of that day == the rounded up current water level **then**
3: save data of the whole day
4: **end if**
5: **end for**
6: return a list of all days when the water level go nearby the current water at the giving current time

Computation Time: is proceeded mainly from calling for-loop on 181 days in all time slots (96 points), similarly a above method it's limited by upper bound of polynomial-time.

4 Test and interpretation of results

In this part, we proceed to test and interpret the results and figures as output of the program. User input parameters to program by running *CLI (Command Line Interface)* as following:

- Input the current water level (in cm): 650
- Input the current time (from 0:00 to 23:00 o'clock): 10.0
- Input the predicting hours (max 12 hours): 8

The first line is the result output in normalized format and the second line is the already converted result in cm. The third and fourth lines are the artificial result calculated based the error difference between history data and data at current time.

```
************************************

The predicting water level is 0.4816616649627397
Result in cm is 519.3341969241858

(Artificial) Result in cm:  650.415850038
(Artificial) Result:  0.640164268486

************************************
```

Figure 3: Result from method 1

```
************************************

The predicting water level is 0.583874466756633
Result in cm is 603.8641840077355

(Artificial) Result in cm:  639.900478532
(Artificial) Result:  0.627449188068

************************************
```

Figure 4: Result from method 2

By comparing the result between method 1 and method 2, we can easily recognize the difference since the result of the first method is calculated by the mean value of all data point at every single time point and the second method chooses the days when their water level were close to the water level at current time. Therefore, user can try both method to get all intermediate and final results and find out the average value or the range where the water level might be.

The next following figures show the visualized results exported out by both methods.

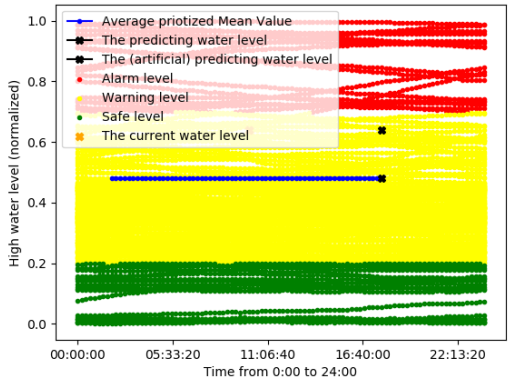

Figure 5: Result from method 1

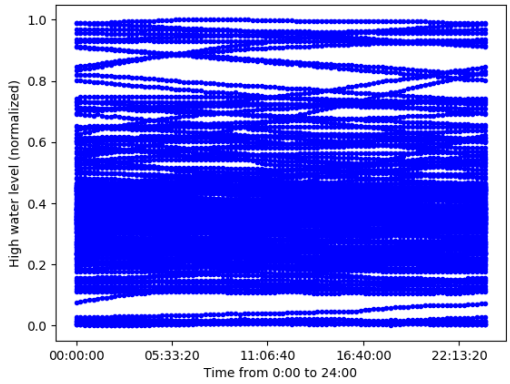

Figure 6: All water level sorted by days

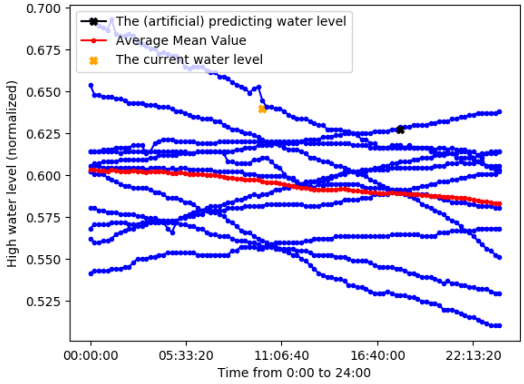

Figure 7: Result from method 2

The blue line in figure 5 is the result calculated by mean value at every point. Data which are used to calculate is beginning from 2:00 AM to 18 AM (due to current time +/- predict time). The mark 'X' at the end of blue line show the predicting water level corresponding to the history data and the above symbol 'X' shows the artificial predicting water level corresponding to the current time.

The blue lines in figures 6 & 7 retrieve all history water data and find out the similarities close enough to the current reality and the red line demonstrates the average value of all blue lines. In figure 7 we easily recognize the worst case and best case scenarios when water level keep rising and decreasing. Hence, for the future prediction we were able to scale the result in the calculated range. In order to determine the correct result closer with the reality situation, it's recommended that user keeps updating with the current weather condition outside. E.g. the weather at current time is having heavily rainfall and weather service informs this situation will last many days, so the trend of water level would be probably rising up and vice versa. Therefore, with this information, we only focus on all the blue lines above the red line with rising up tendency.

5 User Manual

Question 1: How to know whether the current water level is in danger or still safe?

Answer: please see the explain about classification of dataset and Algorithm 1

Question 2: How to help user know the relation between state of water level regarding with location of container?

Answer: Location of Container on the map gives height comparing to the sea level rising (The given data shows about 67 meter). Water lever rising is measured at the PNP (PegelNullPunkt) starts at position 57 meters, so the *'safe'* distance supposes to be x $<$ 67 - 57 = 10 meters. However, user should start concerning as the water level is already in early warning (min of yellow zone) and considering the time cost would take to remove the container, especially in extreme weather condition.

Question 3: In approach 2, how to know that the water level will be rising up or down?

Answer: In reality context, we surely do have data of water level during the last hours in which flooding status have been occurring. Program basically retrieved the water events from history but trend of the current (rising or decreasing) water level depends on the current rainfall and weather situation outdoor. User should keeps monitoring the most updated information from weather service since it has a great impact on flood events and updating the raw-data via Excel file and try to run program simultaneously. Program helps user to navigate status of all possibilities of the water level by comparing values betweens the days in the past and the current day and choosing out the most closest data with the current data. After all user chooses the most possibly best result closest with the reality manually for their decision.

Question 4: How to install program and reproduce the result?

Answer: Program has been developed in Python3 and distributed freely on Github, license Open Source GPL version 2. User can run program by executing Python-script and result will be output via logging-file and in visualized figures. All of input parameters and results will be stored into csv-file and synchronized to database to be reused for another runtime. About the technical detail how to install and run program, please find further information at:

`https://github.com/lphan/utilkit/tree/master/floodpred`

6 Conclusion

Since history experiences proved that exact weather forecast and prediction of natural disaster respectively is extremely difficult, especially when demanding in short time even with help of supercomputer [2] [3] [8]. In this work, we focus on doable scenarios by observing and analysing the dataset in history of water level and find a fast solution to support decision-maker. Furthermore, the program *floodpred* has been developed to answer the question: *'at what water level should user start concerning to make decision?'*. The answer is clearly divided into sub-answers by analysing the data and monitoring the flood water level into 3 different bottom-up layers (green, yellow and red) and hence building different methods to compute and then predict the water level at the expected future time in appropriate reality context. All the calculated results is getting better and more accurate just by keeping the raw-data updated via Excel file without having to change the technical details in computer code. After computation, all the results will be stored into database to be served as cache layer to deliver a fast result in case of computing with the same input parameters. Moreover, the result has been computed in a reasonable polynomial computing time to support a fast decision either.

Finally, program and this document are distributed for the Open Source Community so that everyone can freely join and contribute additional features in the future.

[2] http://www.noaa.gov/media-release/noaa-launches-america-s-first-national-water-forecast-model

[3] https://www.forbes.com/sites/eco-nomics/2012/09/24/7-supercomputers-changing-the-world/

References and Notes

[1] Infoflyer FloodEvac http://www.floodevac.org/uploads/5/2/9/9/52993787/infoflyer_floodevac_engl.pdf

[2] Marc Goerigk, Horst W. Hamacher, Sebastian Schmitt: Decision Support Systems for Urban Evacuation Logistics in Practice;

[3] P. Heler, H.W. Hamacher: Sink location to find optimal shelters in evacuation planning;

[4] Ouyed & Dobler: Chapter 4 - Interpolation/Extrapolation Techniques http://pjl.ucalgary.ca/courses/physics381/computational-physics/Ouyed-Chapter-4-Interpolation-Extrapolation-Techniques.pdf

[5] S. Gopal Krishna Patro, Kishore Kumar Sahu: Normalization A Preprocessing Stage https://arxiv.org/abs/1503.06462

[6] Siva Kumar Subramaniam, Vigneswara Rao Gannapathy, Sivarao Subramonian and Abdul Hamid Hamidon: Flood level indicator and risk warning system for remote location monitoring using flood observatory system

[7] Daher, E.; Kubicki, S. and Guerriero, A 2017. Data-driven development in the smart city: Generative design for refugee camps in Luxembourg, Entrepreneurship and Sustainability Issues, 364-379. http://dx.doi.org/10.9770/jesi.2017.4.3S(11)

[8] Peter Lynch, Meteorology and Climate Centre, School of Mathematical Sciences, University College Dublin, Belfield, Ireland The origins of computer weather prediction and climate modeling, 2007 doi:10.1016/j.jcp.2007.02.034