Flood Warning System

Weixuan Zhang, Ghifari Pradana

Mar 12, 2020

CONTENTS:

CHAPTER

ONE

INTRODUCTION

Solution to the Part IA Lent Term computing activity at the Department of Engineering, University of Cambridge.

The activity is documented at https://cued-partia-flood-warning.readthedocs.io/.

CHAPTER

TWO

INSTALLATION

To run the web interface,

1. Set your Google Maps API key as an environment variable through `bash export API_KEY=<api_key>`
2. Install all the dependencies in `requirements.txt` (a virtual environment is strongly recommended), then run

   ```
   bash plaidml-setup  # set the device used for training
   bokeh serve main.py --port 5100
   ```

   Alternatively, using Docker

   ```
   bash docker build -t floodwarning .
   docker run -it --rm -p 5100:5100 -e API_KEY floodwarning
   ```

   Docker GPU pass through needs to be set according to the specific machine setup used, so it is set to train on the CPU by default. For better performance, run the code without using docker and choose an available GPU using `plaidml-setup`.
3. Go to `http://localhost:5100` from your browser.

CHAPTER

THREE

WEB INTERFACE

3.1 Map

Fig. 1: map

The colour of each point depends on the relationship between the latest water level of that station and its typical range. Hover tool is also implemented, showing useful information when the mouse hovers above a datapoint. Scroll wheel can be used to zoom in and out, and panning can be done by dragging.

Stations can be selected to display its historical water level data either through clicking on the map or using the search box. Fuzzy matching is implemented for the search box. The selected station will be highlighted on the map.

Fig. 2: zoom_out

Fig. 3: search

3.2 Prediction

High risk stations are displayed, and predicted for future changes in water level.

Fig. 4: prediction

The prediction is done both using a least-squared polynomial fit and a recurrent neural network. The LSTM network is trained on data obtained by a sliding window, and is applied recursively to predict future water levels. This is implemented in *floodsystem.predictor* submodule.

As can be seen from the above figure, RNN produces more realistic predictions, especially on periodic and exploding data.

3.3 Warning

When considering the risk of flooding of a town, stations cannot be considered in isolation, so clustering is used on stations whose latest water level is 1.5 times its typical range above the upper typical range. The clustering algorithm implemented is DBSCAN with haversine distance as the distance metric.

Stations within these clusters are considered as high risks, and are shown on the map with less transparency, while the colours indicate relative water levels.

Then the town with the highest relative water level in each cluster is found, sorted using the mean relative water level of the cluster containing it.

Note: The above sections are converted to reStructuredText from README.md using pandoc.

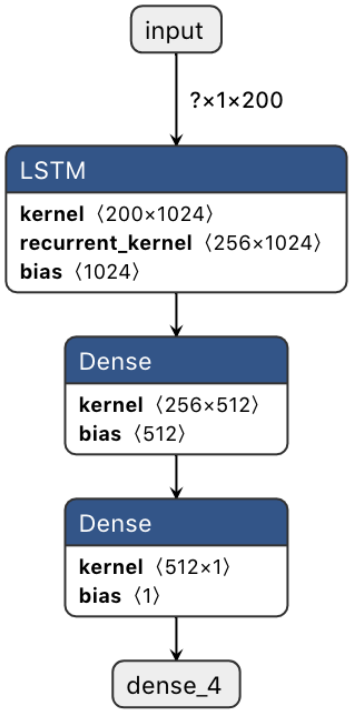
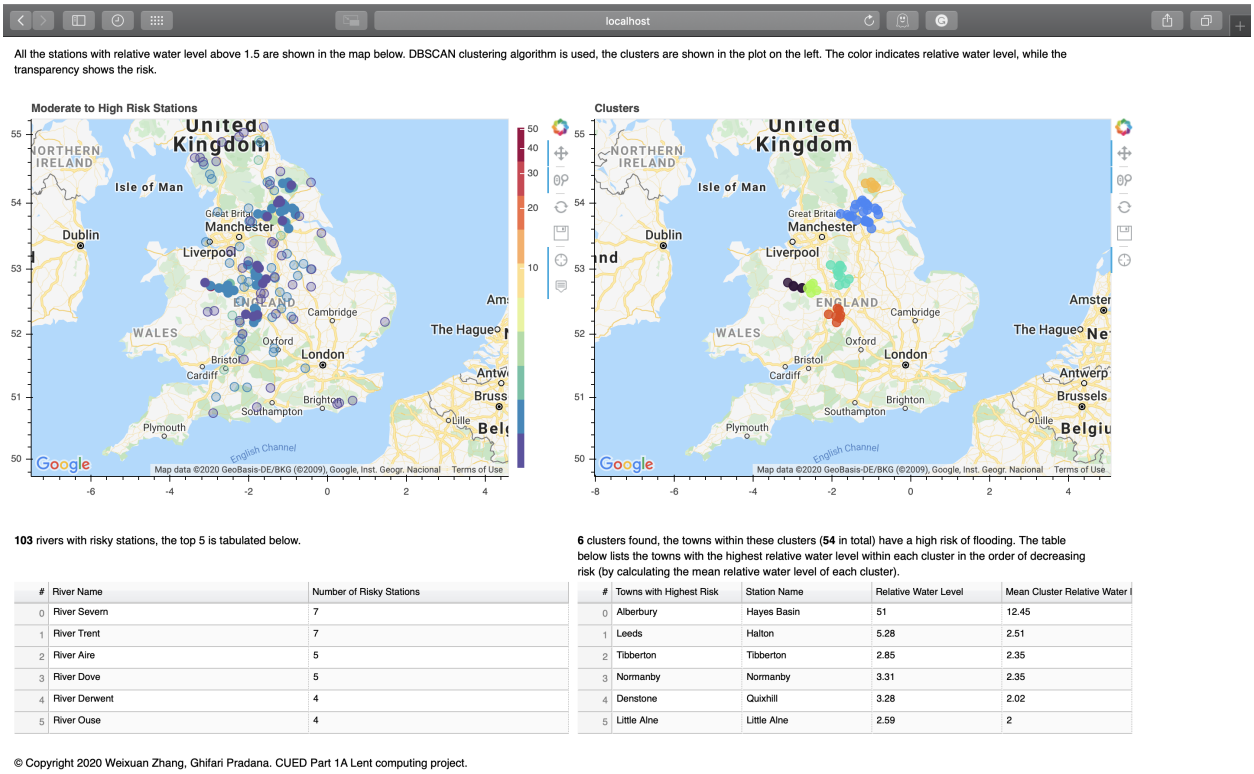


Fig. 5: network



#	River Name	Number of Risky Stations
0	River Severn	7
1	River Trent	7
2	River Aire	5
3	River Dove	5
4	River Derwent	4
5	River Ouse	4

#	Towns with Highest Risk	Station Name	Relative Water Level	Mean Cluster Relative Water
0	Alberbury	Hayes Basin	51	12.45
1	Leeds	Halton	5.28	2.51
2	Tibberton	Tibberton	2.85	2.35
3	Normanby	Normanby	3.31	2.35
4	Denstone	Quixhill	3.28	2.02
5	Little Alne	Little Alne	2.59	2

Fig. 6: warning

CHAPTER

FOUR

FLOOD WARNING SYSTEM

4.1 analysis module

This module contains function for polynomial fitting on data

analysis.**polyfit**(*dates*, *levels*, *p*)

Function that finds the least-square fit polynomial from

Parameters

- **dates** (*list*) – The list of dates for the x-axis.
- **levels** (*list*) – The corresponding water level for each date, y-axis.
- **p** (*int*) – The degree of polynomial that is desired.

Returns Contains the coefficients of the resulting polynomial float: The number of days since the origin of the Gregorain Calendar that was shifted to find the polynomial.

Return type numpy poly1d Object

4.2 datafetcher module

This module provides functionality for retrieving real-time and latest time history level data

datafetcher.**dump**(*data*, *filename*)

Save JSON object to file

Parameters

- **data** (*dict*) – Json string to be saved
- **filename** (*str*) – Path to file

datafetcher.**fetch**(*url*)

Fetch data from url and return fetched JSON object

Parameters **url** (*str*) – url to be fetched

Returns Json string fetched

Return type dict

datafetcher.**fetch_latest_water_level_data**(*use_cache=False*)

Fetch latest levels from all 'measures'. Returns JSON object

Parameters **use_cache** (*bool, optional*) – Whether to use cached data

Returns Json string

Return type dict

datafetcher.**fetch_measure_levels**(*measure_id*, *dt*)
Fetch measure levels from latest reading and going back a period dt. Return list of dates and a list of values.

Parameters

- **measure_id** (*str*) – measure_id of the specified station
- **dt** (*DateTime Object*) – Period of time

Returns Tuple of lists of the form (dates, levels)

Return type tuple

datafetcher.**fetch_station_data**(*use_cache=True*)
Fetch data from Environment agency for all active river level monitoring stations via a REST API and return retrieved data as a JSON object.

Fetched data is dumped to a cache file so on subsequent call it can optionally be retrieved from the cache file. This is faster than retrieval over the Internet and avoids excessive calls to the Environment Agency service.

Parameters **use_cache** (*bool, optional*) – Whether to use cached data

Returns Json string

Return type dict

datafetcher.**load**(*filename*)
Load JSON object from file

Parameters **filename** (*str*) – Path to file

Returns Json string

Return type dict

4.3 flood module

This module contains a collection of functions related to flooding.

flood.**stations_highest_rel_level**(*stations*, *N*)
Function that returns the N number of most at risk stations.

Parameters

- **stations** (*list*) – List of stations (type MonitoringStation).
- **N** (*int*) – Length of the desired list

Returns List of stations(type MonitoringStation)

Return type list

flood.**stations_level_over_threshold**(*stations*, *tol*)
Function that returns stations whose latest relative water level is over some threshold.

Parameters

- **stations** (*list*) – List of stations (type MonitoringStation).
- **tol** (*float*) – The threshold relative water level.

Returns List of tuples in the format (station (type MonitoringStation), relative water level) sorted by the relative level in descending order.

Return type list

4.4 geo module

This module contains a collection of functions related to geographical data.

geo.**haversine**(*a*, *b*)
Function that calculates the haversine distances between two points in kilometers.

Parameters

- **a** (*tuple*) – The coordinate of the first point as (latitude, longitude).
- **b** (*tuple*) – The coordinate of the second point as (latitude, longitude).

Returns Haversine distance.

Return type float

geo.**rivers_by_station_number**(*stations*, *N*)
Function that returns a list of tuples containing the river name and the number of stations it has.

Parameters

- **stations** (*list*) – List of stations (type MonitoringStations)
- **N** (*int*) – The number of desired rivers with the largest number of stations

Returns tuple of (river, number of stations on river) sorted in descending order

Return type list

geo.**rivers_with_station**(*stations*)
Function that, given a list of station objects, returns a container with the names of the rivers with a monitoring station.

Parameters **stations** (*list*) – List of stations (type MonitoringStation).

Returns Set of names of rivers with a monitoring station.

Return type set

geo.**stations_by_distance**(*stations*, *p*)
Function that returns the sorted distances between the input stations and a specified point p.

Parameters

- **stations** (*list*) – List of stations (type MonitoringStation).
- **p** (*tuple*) – Coordinate of the origin.

Returns List of (station, distance) sorted by distance.

Return type list

geo.**stations_by_river**(*stations*)
Function that returns a dictionary that maps river names (the 'key') to a list of station objects on a given river.

Parameters **stations** (*list*) – List of stations (type MonitoringStation).

Returns Keys - river names.

Return type dict

geo.**stations_within_radius**(*stations*, *centre*, *r*)
Function that returns a list of all stations (type MonitoringStation) within radius r of a geographic coordinate.

Parameters

- **stations** (*list*) – List of stations (type MonitoringStation).
- **centre** (*tuple*) – Coordinate of centre in (latitude, longitude).
- **r** (*float*) – Radius in kilometers.

Returns List of stations (type MonitoringStation) within the distance.

Return type list

4.5 plot module

This module contains functions for plotting

class plot.**Map**(*stations*, *origin=(52.207, 0.1131)*)
Bases: object

This class represents a map of stations.

build()

plot.**map_palette**(*station*)

plot.**plot_prediction**(*date*, *data*)
Function that plots the prediction made by predictor.

Parameters

- **date** (*2-tuple*) – List of datetime objects of actual and demo data, list of datatime objects of future predicted data.
- **data** (*3-tuple*) – Lists of water levels of actual data, demo data, predicted data.

Returns Bokeh plot object.

plot.**plot_water_level_with_fit**(*station*, *dates*, *levels*, *p*)
Function that makes a graph of the water level over time for a given station with a least-square fit polynomial with a degree of p.

Parameters

- **station** (MonitoringStation) – The desired station to graph.
- **dates** (*list*) – The list of dates for the x-axis.
- **levels** (*list*) – The corresponding water level for each date, y-axis.
- **p** (*int*) – The degree of polynomial that is desired.

Returns Bokeh plot object.

plot.**plot_water_levels**(*station*, *dates*, *levels*)
Function that makes a graph of the water level over time for a given station.

Parameters

- **station** (MonitoringStation) – The desired station to graph.
- **dates** (*list*) – The list of dates for the x-axis.
- **levels** (*list*) – The corresponding water level for each date, y-axis.

Returns Bokeh plot object.

plot.**plot_water_levels_dynamic**(*source*)
Function that makes a graph of the water level over time for a given station.

Parameters **source** (*type ColumnDataSource*) – The dataset.

Returns Bokeh plot object.

plot.**plot_water_levels_multiple**(*stations*, *dt*, *ncol=3*, *height=250*, *width=300*)
Function that displays a grid of graphs of the water level over time for a given list of stations.

Parameters

- **stations** (*list*) – List of the desired stations (type MonitoringStation) to graph.
- **dt** (*int*) – Number of days.
- **ncol** (*int, optional*) – Number of columns.
- **height** (*int, optional*) – Height of each individual plot.
- **width** (*int, optional*) – Width of each individual plot.

Returns Bokeh plot object.

4.6 predictor module

This module contains functions for the recurrent neural network that predicts furture water levels

predictor.**build_model**(*lookback*)
Function that builds the recurrent neural network, which has 1 lstm layer and 2 dense layers.

Parameters **lookback** (*int*) – The look back value, which determines the input shape.

Returns Untrained model.

Return type Keras model

predictor.**data_prep**(*data*, *lookback*, *exclude=0*)
Function that prepares the dataset by constructing x,y pairs. Each y is determined on the previous <lookback> data points (x).

Parameters

- **data** (*array*) – The water level data.
- **lookback** (*int*) – The look back value, i.e. every y is determined how many x.
- **exclude** (*int, optional*) – The number of latest data points to ignore (default 0).

Returns x. array: y.

Return type array

predictor.**fetch_levels**(*station_name*, *dt*, *return_date=False*)
Function that returns measurements and dates of a specified station since a specified number of days ago.

Parameters

- **station_name** (*str*) – The name of the station.
- **dt** (*int*) – The number of days.
- **return_date** (*bool, optional*) – Whether to return the dates (default False)

Returns

- If return_date False
  - array - Water levels.
- If return_date True
  - list - List of dates (datetime object).
  - array - Water levels.

predictor.**predict**(*station_name*, *dataset_size=1000*, *lookback=2000*, *iteration=100*, *display=300*, *use_pretrained=True*, *batch_size=256*, *epoch=20*)

Function that predict a specified number of future water levels of a specific station.

If the model for that station is not cached, it will be trained according to the parameters specified.

The returned data includes actual data over the specified interval, demonstration data the model produced based on actual data points prior to the displayed actual data, and the predicted date using all the available actual data.

Parameters

- **station_name** (*str*) – The name of the station.
- **dataset_size** (*int, optional*) – The number of days in the dataset (default: 1000).
- **lookback** (*int, optional*) – Look back value (default: 2000).
- **iteration** (*int, optional*) – Number of future water levels to be predicted (effectively the number of times data in passed to the nn) (default: 100).
- **display** (*int, optional*) – Number of real data points to be returned (default: 300).
- **use_pretrained** (*bool, optional*) – Whether to used pretrained model if possible (default: True).
- **batch_size** (*int, optional*) – (default: 256).
- **epoch** (*int, optional*) – (default: 20).

Returns

2-tuple (list, list) - List of datetime objects of actual and demo data, list of datatime objects of future predicted data.

3-tuple (list, list, list) - Lists of water levels of actual data, demo data, predicted data.

Return type tuple

predictor.**train_all**(*stations*, *dataset_size=1000*, *lookback=2000*, *batch_size=256*, *epoch=20*)

Function that trains models for all station supplied.

Parameters

- **stations** (*list*) – List of MonitoringStation objects.
- **dataset_size** (*int, optional*) – The number of days in the dataset (default: 1000).
- **lookback** (*int, optional*) – Look back value (default: 2000).
- **batch_size** (*int, optional*) – (default: 256).
- **epoch** (*int, optional*) – (default: 20).

predictor.**train_model**(*model*, *x*, *y*, *batch_size*, *epoch*, *save_file='./floodsystem/cache/predictor_model.hdf5'*, *show_loss=False*)

Function that trains and saves the Keras model.

Parameters

- **model** (*Keras model*) – The built model.
- **x** (*list*) –

  x.
- **y** (*list*) –

  y.
- **batch_size** (*int*) – Batch size.
- **epoch** (*int*) – Number of epochs.
- **save_file** (*str, optional*) – Path to save the trained model file (default: './flood-system/cache/predictor_model.hdf5')
- **show_loss** (*bool, optional*) – Whether to display the loss-epoch graph after training.

Returns The trained model.

Return type Keras model

4.7 station module

This module provides a model for a monitoring station, and tools for manipulating/modifying station data

class station.**MonitoringStation**(*station_id*, *measure_id*, *label*, *coord*, *typical_range*, *river*, *town*)

Bases: object

This class represents a river level monitoring station

latest_level
The latest water level of the station

Type float

property coord
Coorinates of the station in (latitude, longitude)

Type tuple

property measure_id
measure_id

Type str

property name
Station name

Type str

relative_water_level()
This method returns the latest water level as a fraction of the typical range.

Returns 0.0 (corresponds to a level at the typical low) to 1.0 (corresponds to a level at the typical high)

Return type float

property river
River name

Type str

property station_id
station_id

Type str

property town
Town name

Type str

property typical_range
Typical water level range of the station (typical_low, typical_high)

Type tuple

typical_range_consistent()
This method checks whether the data it receives about the typical ranges are consistent(That data is available and the low range is lower than the high range).

Returns Returns whether or not the data is consistent

Return type Boolean

station.**inconsistent_typical_range_stations**(*stations*)
This function checks takes in the list of stations and checks to make sure the typical range for each are consistent.

Parameters **stations** (*list*) – List of stations (type MonitoringStation).

Returns List (type String) of all the stations with inconsistent typical ranges in alphabetical order

Return type list

4.8 stationdata module

This module provides interface for extracting station data from JSON objects fetched from the Internet and updating their water levels

stationdata.**build_station_list**(*use_cache=True*)
Build and return a list of all river level monitoring stations based on data fetched from the Environment agency. Each station is represented as a MonitoringStation object.

The available data for some station is incomplete or not available.

Parameters **use_cache** (*bool, optional*) – Whether to use cached data

Returns List of stations (MonitoringStation Object)

Return type list

stationdata.**update_water_levels**(*stations*)
Attach level data contained in measure_data to stations

Parameters **stations** (*list*) – List of stations (MonitoringStation Object)

4.9 utils module

This module contains utility functions.

utils.**sorted_by_key**(*x*, *i*, *reverse=False*)

For a list of lists/tuples, return list sorted by the ith component of the list/tuple,

Examples

Sort on first entry of tuple:

```
>>> sorted_by_key([(1, 2), (5, 1)], 0)
[(1, 2), (5, 1)]
```

Sort on second entry of tuple:

```
>>> sorted_by_key([(1, 2), (5, 1)], 1)
[(5, 1), (1, 2)]
```

Parameters

- **x** (*list*) – A list of lists/tuples to be sorted
- **i** (*int*) – Sort according to which element of the inner lists/tuples
- **reverse** (*bool, optional*) – Default False

Returns Sorted list of lists/tuples

Return type list

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

INDEX

T

U