Ma422 - Project report

The cloud dataset

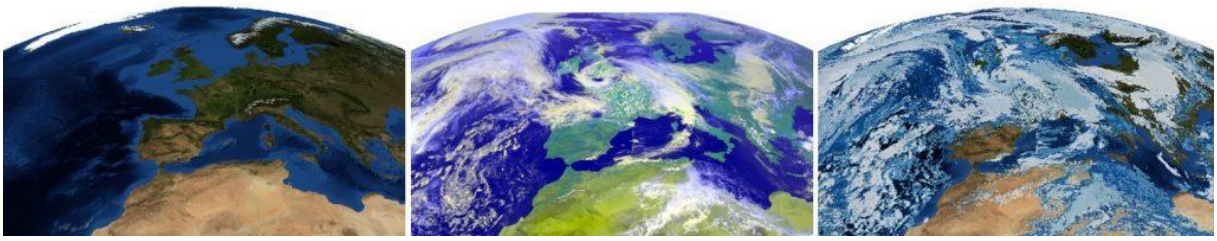

Table of contents

Introduction

Preamble

The following report is divided in four parts, each tackling a different step of the project, sometimes unsuccessfully. The *Part IV* can be seen as a good summary of all the work done beforehand, as it features most of the conclusions of the previous parts, in their lightest form (both from a file weight and computation cost perspective). **It is thus recommended to fully run only this part's script.**

All along the report, figure referenced as *[i]* can be found in the project folder, as *[i]_name.png.*

Please note that each scripts offer the option to **precise the absolute path to the *Project* folder**, as *abs_path*. This path should be changed accordingly for the scripts to work on your computer.

Please note that scripts may call a module named ML.py, which regroups the different algorithm.

The cloud dataset

The cloud dataset is a sample of earth pictures taken from space by the EUMETSAT and projected on a 728x728 frame, centred on Western Europe. The data is sampled every 15 minutes over 2017 and 2018 (cf. figure [1]) and is saved as NumPy arrays. Instead of representing the clouds as coloured pixels, each dot is a labelled cloud type, as followed :

Labels 1-3 :	No clouds or no data
Labels 4-8 :	Very low, low mid, high, or very high clouds
Labels 9-13 :	Fractional clouds or semi-transparent very high

PART I- Editing the dataset

Presentation

The dataset weights 30GB and is stored as suboptimal float NumPy arrays. The goal being currently to train two versions of the learning algorithms, one recognizing clouds by their image and the other by their shape, one training example had to be split in 13 arrays twice, to train each algorithm on each label. The dataset is thus too heavy and has to be compressed.

Creating the 26 subarrays

Every array is divided in two categories with 13 labels each by the *compute_mask_and_border()* function. Label values ranging from 1 to 13 are normalized as 0 and 1 pixels, as the cloud type is now specified alongside a 3rd dimension *label* axis created by the concatenation of the arrays.

To obtain the border arrays, the dataset follows the same treatment and is then convoluted by a 3x3 kernel using the SciPy library. Plots of the results are generated by *plot_mask_and_border()*, as shown by figure [0].

Saving the 26 subarrays

Every array is now 26 times its original weight, and three dimensional (label, x, y). Because of how heavy each array still is, the data had to be batched along a 4th dimension, *time*. Every batch of 96 arrays is saved as one netcdf files, thus resulting in one dataset per day (4 samples per hour, for 24 hours), named as followed : *{year}M{month}[…]{day}.nc*.

The data is saved as int8 and compressed. From 15GB of data in 2017, the 26 times larger new dataset weights now less than 13GB, in two folders : *TheCloudDataset_labelized_arrays_train* and *TheCloudDataset_labelized_border_train*.

PART II- Testing the machine learning algorithms

Presentation

Data is now organized as 4 dimensional netcdf files, and can be easily loaded and exploited. We'll now perform a Principal Component Analysis on the dataset to reduce the size of every array. Then, we'll test the OneVsAll algorithm and create a neural network over the 2017 data, to then test it on the 2018 samples.

PCA

PCA was performed with k=10, 20 and 30 main components, and saved in their own folders. Because of the length of the dataset*, compute_pca_dataset()* loads and reduces only one array over 8, thus creating a smaller dataset, as if sampled every 2 hours. PCA data is then flatten as a vector of size 728*k, to match the shape required by the training algorithm. Examples of the PCA output and the recovered array is shown in figures [1b] and [1c].

OneVsAll Algorithm

Although failing continuously over labels 2, 3 and 8, due to the lack of data to train with (arrays of these labels are mainly empty), the *OneVsAll* algorithm shows very good results when tested over the trained dataset (cf. figures [2] and [3]). However, success rate nosedives when tested over the 2018 dataset, thus resulting in an overall failure (cf. figures [4] and [5]).

Thetas trained over the year are saved in *Project* as *trained_thetas_k{main_components}[…].npy.*

Neural Network

The same tendency is conserved when training the one-layer neural network over the whole 2017 PCA dataset, with even worse results, both for the 2017 original training source (cf. figures [6] and [7]) and the 2018 testing dataset (cf. figures [8] and [9]).

Trained neural network thetas can be found in *Project* as *trained_NN_k{main_components}[…].npy.*

PART III- Perfectioning the algorithms

Presentation

After the relative failure of the previous part, it was decided to reduce the ambition of the project before going on. As the *OneVsAll* algorithm seemed to be not working at all on the 2018 dataset, we decided to emphasize our efforts on the neural network. Because of computation time and memory allocation issues, we decided to fully regenerate the dataset as sliced arrays, to keep 128x128 frames and perform PCA on it again, using *Score Z* normalization.

After analysing more deeply the recurring cloud shapes as shown in figures [10] and [11], categories were also reduced to consider clouds as larger ensembles (cf. figure [12]). From now on, these new datasets fit every array in only one netcdf file named *new_array[..].*nc, which can be found in *Project/P3_Training* (resp. *P3_Testing*).

Improved neural network

Nevertheless, results are still not satisfying enough, and some overflow errors appear as some zeros arrays are not correctly handled, resulting in the same poor results as in *Part II* (cf. figure [13]).

On the other hand, the assessment is largely better when testing the border dataset, as the trained dataset gives excellent results. Yet the testing dataset is plain bad, just like the *OneVsAll* example (cf. figure [14]).

PART VI- Results and development

Presentation

This final part is a new start from all what was learned previously and is entirely dependent of the *Project/small_cloud* directory. The *data.zip* file should be emptied in the same directory. Data is 128x128, normalized, and will be subjected to k=10 and k=30 PCA.

Pixels repartition and data merging

Pixels are distributed in unequal categories, as shown in figure [15] and are now regrouped in slightly more balanced groups (cf. figures [16] and [17]). Data is thus merged accordingly in 4 dimensional netcdf files.

Neural Network 2

The thetas of this new version of the neural network computation are calculated on the k=30 PCA datasets, as the computation cost is lighter than its *OneVsAll* counterpart. However, we've tried using the SciPy normalization algorithm, and the results are actually kind of worse than before (cf. figure [19]). The merged version, using both image and border dataset concatenated (combined) together doesn't give any satisfying result either (cf. figure [20]).

OneVsAll 2

Due to computation time (4min/label for k=20 and 25min/label for k=30), both image and border datasets use a k=30 PCA (cf. figures [21] and [22]), whereas combined *image&border* dataset uses two k=10 PCA datasets concatenated together (cf. figure [23]).

The same fault shown in *Part II* remains, mediocre when tested over the learning dataset (top lines results), and heavily biased towards *label 0* (bottom lines result), as the normalization algorithm seems to be the main weakness of the learning process.

Final review of the results

Overall, the *OneVsAll* algorithm results in *Part IV* are slightly better than those obtained in *Part II.* When it comes to the Neural Network training, the most advanced version is the one created in *Part III.* Ideally, *Part III* results and datasets should be polished using techniques from both parts *III* and *IV.*

Conclusion

Considering the limit of time and the relative results obtained, the project had to stop here, although there's still plenty of room for improvement, notably by trying more powerful learning tools, algorithm, and testing more suitable normalization processes. Furthermore, the next step of the project would be to be able to directly identify some cloud types by computing the K-mean algorithm (cf. figure [24]) over a raw EUMETSAT image (cf. figure [25]), to generate 4 to 13 labels, and then try to use the previously trained learning algorithm to identify clouds from scratch.