Simon Fraser University

CMPT 318 Data Science

Project Report

Jagrajan Bhullar

Duc Phuong Nguyen

August 8, 2017

1 The Problem

The main idea behind this project was provided by our instructor, Greg Baker. The idea was to see if we could automate the process of categorizing weather by observing still images, instead of having people manually categorize the weather in still images.

This idea translated into our project as the following question: can a machine be trained to correctly determine the weather observed in an image? More specifically, all our training data and testing data was taken from a camera at a fixed location, so the question turned into, could a machine correctly distinguish different weather conditions in an image? This question remains unclear, and raises even more questions. What kind of weather conditions? How broad or specific are the weather categories? How many different weather conditions can a single image have? In this report, we will find out that these questions will help explain some of the limitations with trying to categorize weather using a machine. Later on, we will discuss how we refined the question after our inital results were limited in accuracy and consistency.

2 The Data

Our data was provided by our instructor, Greg Baker. As mentioned in the Project Details, the weather data we have (from YVR airport station) does not perfectly match the weather images (from English Bay) so some clean-up jobs were required before training models.

First, weather data (observation) was read as a DataFrame which contains two main columns (Date/Time and Weather). Rows with empty observations will be excluded and Date/Time string will be converted to *datetime64* object (using *pd.to_datetime()*) for merging later on.

Next, the same process was implemented with images data from KatKam. The original image scale was quite big (256 x 192 = 49192 inputs) and more than necessary for training models so it was re-sized to only half (192 x 96 = 12288 inputs) while not losing any accuracy at later steps (tested).

Each pixel was then represented as a *float64* value (instead of 3 distinct integer values for *RGB colors*, to limit the number of input features). Finally, they were reshaped to fit in the DataFrame with 'datetime' and all the features as columns and merged with the weather data to form a new DataFrame to be used for learning.

3 Trying Out Classifiers

Our first intuition would lead us to a classification problem where inputs are the images data and outputs are the weather description, so let us start with some simple methods to deal with this and to produce some kinds of result.

3.1 Single-Label Classifier

My first attempt to solve this problem is to train a model using single-label classifier (where different set of labels will give different outputs; for example, label set [Rain, Cloudy] would be distinguished from [Rain] or [Cloudy], which is not very practical, but worth trying).

Three approaches were used to build the model pipeline for predicting outputs: *KNeighborsClassifier* (with n_neighbors = 5, 7, 9, 11), *DecisionTreeClassifier()* and *RandomForestClassifier* (with n_estimator = 10, 20, 30). Although the results were not very surprising, they all gave different accuracy scores as followed: (approximately) **0.43** for *kNN*, **0.25** for *DTree* and **0.18** for *RForest*. In addition, *PCA(250)* was used to decrease the number of dimensions in the inputs so runtime can be varied from 20 to 30 seconds, depending on parameters used.

3.2 Multi-Label Classifier

However, this is actually considered a multi-label classification problem where the output can (and usually) belong to more than one categories (for example: [Drizzle, Fog] must be assigned two distinct labels Drizzle and Fog, instead of just one label). This led us to training a model using *OneVsRestClassifier()* with built-in estimator (such as *kNN*, *DTree* and *RForest* as mentioned above) to improve the accuracy score.

Since the original weather descriptions were in string format, we need to perform some string processing to get desired outputs: first, separate them commas (,) and then, put them in list. These lists were then converted into a 2-D binary indicator array where element in row i, column j indicate the presence of label j-th in sample i-th, using *MultiLabelBinarizer()*.

The rest of the jobs is to run it through the model and get the results as below (estimated):

Number of categories: 17

KNeighborsClassifier(n_neighbors=9)	**0.33**
DecisionTreeClassifier()	**0.16**
RandomForestClassfier(n_estimators=20)	**0.12**

3.3 Multi-Label Classifier with reduced categories

The main issue here might lie in the number of categories we have, in relevance to the weather descriptions. It could be a major obstacle for the classifiers to do a good job. Moreover, the images data obtained is not necessarily perfectly aligned with the weather observations (since they are recorded at different location and time) and it is very hard for the camera to capture all the information, merely by taking pictures of the sky (visually, there is little to none change in the images between types of Rain or Snow, or even between Rain/Snow and Cloudy).

Thus, it could be a good idea to group up similar weather conditions and form a new set of labels to help the classifier identifying the outputs more robustly. Below are list of weather description before and after using filter:

Table 1: List of weather conditions

Original	**Reduced**
Clear Mainly Clear	Clear
Cloudy Mostly Cloudy	Cloudy
Fog Freezing Fog	Fog
Drizzle	Drizzle
Rain Heavy Rain Moderate Rain Moderate Rain Showers Rain Showers	Rain
Snow Snow Pellets Snow Showers Moderate Snow	Snow
Thunderstorms	Thunderstorms

The final results acquired are not too terrible, indeed! Using the same methods we have discussed last section, we can reach up to **0.62** accuracy score, equivalent to a **90%** boost while reducing number of categories to only half.

Number of categories: 17

KNeighborsClassifier(n_neighbors=9)	**0.62**
DecisionTreeClassifier()	**0.35**
RandomForestClassfier(n_estimators=20)	**0.33**

Of course, this does not compensate for the lack of details in the outputs since it is a trade-off and we probably need to tackle the problem in more efficient ways.

4 A Different Approach

The increase in accuracy by limiting the categories was great to see, but it felt there was still lots of room for improvement. We decided to approach this problem differently to see the results we could get. Up to this point, we were asking a machine to classify images based on the weather that was observed. But was this the right question? Looking at images, even for us humans, it was difficult to tell what exactly was the weather condition, especially with the low resolution of the images. We found that we could not run the machine with higher resolution images, because the scaled images took far too long to run, so anything of a higher resolution would make things even worse.

So we could not brute force for better results, mainly because our computers were not powerful enough to perform quick computations. Instead, we refined our approach. What if instead of asking a machine to categorize images based on the weather present, we asked the machine to check whether or not a certain weather condition was present? We could give the machine an image and ask, are there clouds? Is it raining? Is it foggy?

5 Single Weather Condition Results

Running the machine when looking for a single weather condition looked very promising at first. The scores were much higher than any classifiers we had used when trying to categorize images from into seven different options. However there is a huge tradeoff here, as we can only test for one weather condition at a time.

5.1 Predicting the Clouds

The first attempt at checking for individual conditions involved clouds. In order to determine whether a picture was cloudy, we should simply check if the observation included "Cloud." Running this test a few times, we found that the nearest neighbours classifer once again proved to be the most accurate with posting scores in 0.70-0.75 range. However, after looking at the data set, it can be noted that not all images with clouds were properly flagged as being cloudy. In order to get around this, we considered any weather conditions that cause precipitation to also be cloudy. With this modification, the accuracy improved and was consistently above 0.90.

However, there were still some images left that had an incorrect cloud flag in the filtered data set. The final work around was to give a clear flag to those images with "Clear" in their descriptions. Any image that had a matching "Cloudy" flag and "Clear" flag were removed. This seemed to have no effect on the accuracy score, as the score remained around the same range.

5.2 Predicting the Rain

The next condition we tried to get a machine to predict was rain. Rain was difficult to tell with a human eye on a low resolution image, so were not feeling optimistic about this test. However, after running the machine, the results gave an average score of about 0.80.

5.3 Predicting the Fog

Fog felt even more hopeless than rain. How was a machine supposed to be able to tell if an image had foggy conditions. Fog is a near invisible layer that can be hard to tell even by the human eye. However, we let the test run, and were shocked when we got a score of 0.95 consistently.

This looked too good to be true, and we soon discovered that this indeed was the case. Only 119 of the 2248 images were categorized as foggy. So even if the machine guessed “no fog” every time, the score would be above 0.95. Furthermore, this can easily be the case when using the nearest neighbours classifier where almost every point is going to be “no fog.” Trying this test out with other classifiers produced far worse results, so fog was something that we could not reliably predict, other than assuming there is no fog and hoping for the best.

6 Meaningfulness of Results

Our different approach definitely increased our accuracy scores, but does that really mean anything? If a machine says there are no clouds, but also says that it is raining, is that a good prediction? No, because these a very contradictory predictions, and we cannnot assume that one certain weather condition being predicted correct means others will do the same as well. This becomes a case of when we want to use product rule of probability to compute the true accuracy of our machine. If we go by the scores mentioned above about predicting clouds, rain and fog, then the probability that a machine makes a correct prediction is $0.90 \times 0.80 \times 0.95 = 0.68$, which gets very close to the test score of categorizing images into seven conditions.

Even if we were able to magically bump up the accuracy for the seven categories above, would our results still be meaningful? If our machine correctly predicts rain, does that mean anything? Rain could range anywhere from a light drizzle to heavy showers.

Our problem was inspired on the idea of whether we could have a machine describe weather conditions instead of having to manually input everything. Is it worth it if a machine can make a correct prediction only 60% of the time, and that is with a very limited set of categories?

We feel that with the machines we have created, and the tests we have run, this is not a solution to our original problem. Our limitations in the areas of machine learning, image manipulation, inconsistencies in data, and lack of powerful hardware all played a role in limiting our results. So for now, we believe that a machine is not an ideal solution to replace manual input of weather conditions.

7 Accompishment Statements

Webcam, Weather and Predictions - *a data science task to identify weather conditions based on camera photo shoots*

7.1 Duc Phuong Nguyen

- Coordinated with a groupmate to come up with several effective approaches to divide and tackle the main problem.
- Analyzed and reorganized acquired data from different sources to create a meaningful and interactive set of data to be used.
- Built and tested three separate models for machine learning using Python (with Scikit-learn) to improve prediction results and boosted accuracy score by 90%.

7.2 Jagrajan Bhullar

- Collaborated with a group mate to tackle a conceptually challenging problem.
- Tried out out-of-the-box approaches towards tackling a problem that seemed to have an upper limit on accuracy, using Scikit-learn and various classifiers. The purpose was to see if an unorthodox approach could create a breakthrough that classifiers could not.
- Created a report that was organized and straightforward to follow using LaTex.