Study on Data Analyzation in Image Recognition and Weather Prediction

CMPT 318 Final Report
December 3rd, 2017

Liangze Hao
Haoyu Liu

Abstract

The aim of the study on Big Data in Image Recognition and Weather Prediction is to helping releasing of labor force and improving the efficiency of labeling weather. Our purpose is to find out a best model to predict the weather condition in Vancouver based on Kat Kam image from Downtown Vancouver.

Weather data are retrieved from YVR airport weather station and images are taken by webcam in Downtown Vancouver. After applying data cleaning, statistical analyzation and image processing techniques, we finally find out a satisfiable model which can be trained to predict weather.

Introduction

Data analyzation is an modern and efficient way of solving our daily problems, or releasing labor force. My group mate and I realized this technique can be applied to “weather labeling” work. Currently, weather stations are using human to label weather at every checkpoint. However, this costs a lot of labour force and the result is not accurate enough. We can change this situation by building a model which takes an webcam image as input and automatically label that image with its predicted weather. Our goal is to build such model to predict the weather and improve the accuracy.

Methods

Preparing Data

Images are retrieved from Kat Kam and resized to 256×192. Images are taken by a steady camera, from 2016.06.05 to 2017.10.31, hourly. By going through these photos, we first realized there are some dark photos. Since those images can not show its weather condition when it is too dark, we considered such images as outliers and discarded those images. And we drop bottom part of those images, take only first 120 rows of pixels. Because compare to cities, roads and forest, the pixel values of sky is more valuable.

Weather information is downloaded from YVR weather station. They show weather labels, temperature, humidity, etc, as same period of time as photos. Because the weather station is away from camera, using temperature and humidity is not accurate to predict. And our main purpose is use only images to predict the weather. Then my teammate and I decided to extract only the weather labels.

By counting those labels, we figure out that ‘Clear’, ‘Cloudy’, ‘Mostly Cloudy’, ‘Rain’ and ‘Snow’ are significantly more than other weathers. It is clear that only these weathers are reasonable to be considered.

The weather station seems to collect weather conditions every 3 hours. So about 2/3 of the weather labels are unlabeled. Our first attempt was to discard all these unlabeled ones and train our models with only labeled ones to see accuracy. We got more than 1400 labeled examples.

First Iteration

Our plan for our first iteration was to use cv2, open python library, to read RGB value of each picture and put all the RGBs into a spark dataframe. Then use basic machine learning models to train and test the result of it.

During experiment, we met a problem that data size exceeds the memory restriction. In order to make data usable, we have to put all rgb values of each image in one row. In this case, each row contains 256x120x3 = 92160 features. This exceeds most computers memory size and we have to work around.

Second Iteration

In order to load data into data frame. We decided to resize each image into a small scale, 30x20 using cv2 resize function with inter nearest function. We discovered that even cv2 is an efficient choice to read images, we still need a lot of time on reading images since we have over 6900 images. To avoid unnecessary time cost on repeat reading images when testing, we decided to read images and output a CSV file containing all RGB values first, then analyze these data in another program. New scale works perfectly on local PC and we can put those value into a dataframe.

We first load our CSV file (containing all 30x20 images RGB) into pandas dataframe. Each row contains all RGB values of each picture. Then we merge this data frame with a relevant weather column to make a full dataframe. RGB values are X and weathers are Y. This full data frame was splitted into training set and testing set randomly with training testing ratio 3:1. We used training set to build Naive Bayes, KNN and SVM models and used test set to get the score of each model.

When doing researched about KNN models and SVM models with different parameters, we decided to use N-Neighbour between 1 and 100 since the local optimal result will appear in this range. And we decided to use C value between 0.1 and 100. And base on the computer performance on both of our computers, 20 times of random split costs reasonable time for use to run. So we splitted and trained 20 times for those models. The average score of Naive Bayes model was 0.437. The best average score of KNN was 0.533. The best average score of SVM was 0.475.

Analyzing our score, we conclude our model is not satisifiable.

According to our observation, we suggested two methods to improve accuracy and to improve calculation efficiency. First method was to apply PCA to make this high-dimensional data space becomes a lower dimension data space. We assume this can improve our prediction since it eliminates unrelated details, as well as boost our calculation efficiency since it results a smaller number of features to be calculated. Second method was to fill unlabeled datas to increase the amount of training data.

Third Iteration

Our first task was to fill in missing weather labels. By checking some images of unlabeled time point, we came up a idea about how to fill in those labels. We observed that the weather condition is almost same as its neighbours labels. That is, if we know that it is

rainy at 2PM and 5PM, then it is very likely that it is rainy between 2PM to 5PM. Base on this assumption, we try to use the adjacent weather labels of an unlabeled image to give it a weather label. Here we add a set of new features to these data as 'Likelihood' for each weather condition. The default values of likelihood is 0 and will set to 1 if there is a real weather label of that likelihood at that time point. We calculated the likelihood of each weather w for unlabeled time point t by following function:

$$P(t,w) = ½*P(t+1,x)+ ½*P(t-1,x)+ ¼*P(t+2,x)+ ¼*P(t-2,x)$$

where t+1,t-1,t+2,t-2 are t's neighbour time point. After applying above function to generate likelihood of weather for unlabeled time point, we assign the label with weather of highest likelihood at that time. For those time point which have equal probability of two weathers, we discard them. To validate if this method shares same weather distribution with original values, we counted each weather and concluded that they have same frequency of occurrence. We call these labeled examples "simulated example". We got more than 6000 labeled examples after this operation.

Then we put all these data into pandas dataframe, and used PCA function in sklearn to decrease the dimension of data. After testing on several values on number of dimensions. Our PCA dimension was decided to be 250. Data are randomly splitted into training set and test set. For data in test set, we discard all "simulated examples" to keep the score reliable. Then we repeated same operation as in third iteration. The average score of Naive Bayes model was 0.354. The average score of KNN was 0.63. The average score of SVM was 0.425. We observed that no matter what C value we decided for SVM, the score will not be changed.

According to our observation and our statistic knowledge we conclude that Naive Bayes is not a good model for this problem. KNN is improved but the result is still not good enough. SVM keeps returning same result, which means there is problem when we applying this model. After researching, we know the reason is SVM can be easily overfitting when it uses 'rbf' kernel with high gamma value.

To improve our problems, we need to further clean our data. We went through those incorrect predictions, and we realized that all snow weather are incorrectly predicted in KNN model. We found the reason was there are only 133 snow weathers in more than 6000 records. In this situation, machine cannot distinguish snow accurately. Also there is no absolute difference between snow and rain shown by image. Then we checked images of label 'cloudy' and 'mostly cloudy', there is only slightly differences between these two types of images. Since the weather station and camera are far away, we can assume that these two weathers are not accurately recorded. Then we can consider both of these weathers as "Cloudy". As a result, we concluded that only 3 weathers are available to be analyzed in this situation: Clear, Cloudy and Rain.

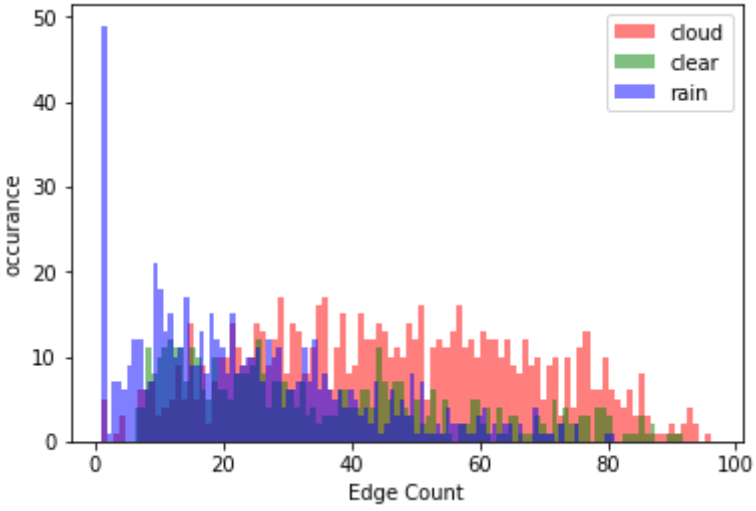
Fourth Iteration

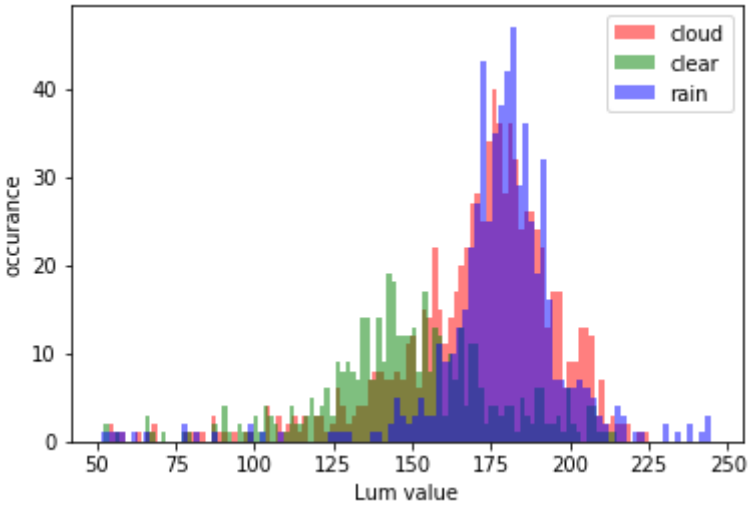
Until this state, we realized the important factors that affects the prediction: information from image, amount of training data and differences between results. In order to improve

our prediction, we were going to increase the number of features from images. We first increase the sampling rate. Instead of resizing the original image into 30 x 20, we resized it into 128 x 80. Thus PCA can return better features to distinguish different weather conditions. Since we do not need to worry about reading images, the only cost of larger sampling rate will be reading time of a larger csv file which is acceptable. My group mate and I also did some research about image processing and we believe edges and luminance will help us to improve our prediction.

We went through those images and realized the difference between clear and cloudy is the amount of edges in sky. The main difference between clear and rain is the luminance of sky.

First we checked the different values of edge numbers and luminance among different weathers : clear, cloudy and rain. Since each weather contains enough amount of examples, we can do a levene test and then a T-test to see the difference of these weathers. Following images show the different edge amounts and average luminance value among different weathers.





Although those T-test fails, we can not say there is a difference in mean of these values, we can still see the difference between edge counts of Cloud to edge count of other

weather and difference between average luminance of clear to luminance of other weather. After observing above graphs, we believe add these two features will improve our model. Then we change the program which produce CSV file and adding two features: edge counts and average luminance to the CSV file.
Then we loaded all rgb data into data frame and used PCA to change data into 250 dimensions. We merged average luminance and edge counts to the output of PCA and got a X with 252 features. Then we do same split process.

We tested it with different n neighbour in range [1, 100] to get the highest score. Recorded highest score with relevant n number. For each time of SVM test, we tested svc model with C in range [0.1,100], gamma in range [1e-10, 1e-6]. The average score of Naive Bayes model was 0.466. The best average score of KNN was 0.696. The best average score of SVM was 0.728.

Conclusion

In the end, we are able to build a model to predict weather of clear, cloudy and rain based on Kat Kam image with accuracy around 0.73 which is acceptable. By applying this model, we are able to label a weather condition to an image of Kat Kam webcam in Downtown Vancouver. However, there is still lots more we can do. If we collect more data about snow in Vancouver, then we may be able to predict with 4 different outputs. So further goal of this project may be improving on number of outputs.

Liangze Hao's Accomplishment Statements:

Generate dataframe by reading and analyzing weather report and images.
Clean data by applying different functions to improve the performance of models.
Design approprite testing method to decide best of models.
Implement the program based on testing result to show the best model we got.
Adding more details of our experiment to the report to show our progress better.

Haoyu Liu's Accomplishment Statements:

Built initial models of Naive Bayes, KNN and SVM using sklearn library to collect result data , which can be used to analyze problems

Improved model by adding edge attribute based on knowledge of image processing and results in 8% increment in accuracy

Figured out solution to solve overfitting problem in SVM collaborating with teammate to make SVM model available for big amount of data

Collected test data and analyzed to suggest possible solutions to improve current models and used charts to help analyzing