1 Problem

Given two sets of data: images and weather reports, the problem is to train a model to accurately guess the weather from the images. The five weather categories used are ”Clear”, ”Somewhat Cloudy”, ”Very Cloudy”, ”Rain”, and ”Snow”.

2 Data

I used the given data. roughly ten thousand weather reports, and five thousand images. First I collect the weather data, which is needed to be in a precise format: csv files with data starting on 15th line of text. Right away I clean it, then from the Date/Time column I choose which image files to read from. I read with 64x48 pixel resolution to speed along the process.

In cleaning I make the following decisions.

- Use only good ”Data Quality”, (which is not missing or estimating data)
- Exclude any weather that is ”NA”, since it is not useful
- Exclude thunderstorm data since the sample is too small
- Ignore ”Drizzle” weather since it confuses me, (18 rows only)

I end up with 2200 matched weather-image pairs. Classifications are extracted from weather, while features are extracted from the image pixels. From weather I extract two categories of information: The sky categorized as ”Clear”, ”Somewhat Cloudy”, ”Very Cloudy”, ”Rain”, and ”Snow”. And fog, categorized as ”Fog” or ”No Fog”. At this point I have only used the first 5 class sky category. As for features, options I’ve made available are the overall Brightness of the entire image, the overall red green and blue, and a list of all pixels as grayscale or as r,g,b triples. I used overall Brightness, RGB, and grayscale pixel map. I ignored all pixels as rgb, because it caused the program to run slowly and seemed to offer no performance boost.

Other feature extractions are possible, and this is likely where I would continue this project. Specifically I had my eye on contouring, which I believe could yield better results.

3 Analysis

Because there were so many features I started with decomposition. The techniques I used for each model were

- Scaling (with resolutions of 256*192, 128*96, 64*48)
- PCA (with feature counts within the range of 10 and 250)

Then I used the following models.

- Naive Bayes
- K-Nearest Neighbours (slight performance boost with n=10)
- Support Vector Machine (kernel radial based function)

I also tried to use ONLY the overall brightness and colours, which did surprisingly well with K-Nearest-Neighbours.

I found that my best results were with Support Vector Machine, C=2, gamma=1e-6. Resolution and PCA did not make a big difference when I used high values, so I stuck with 64*48 and 100 features.

4 Results

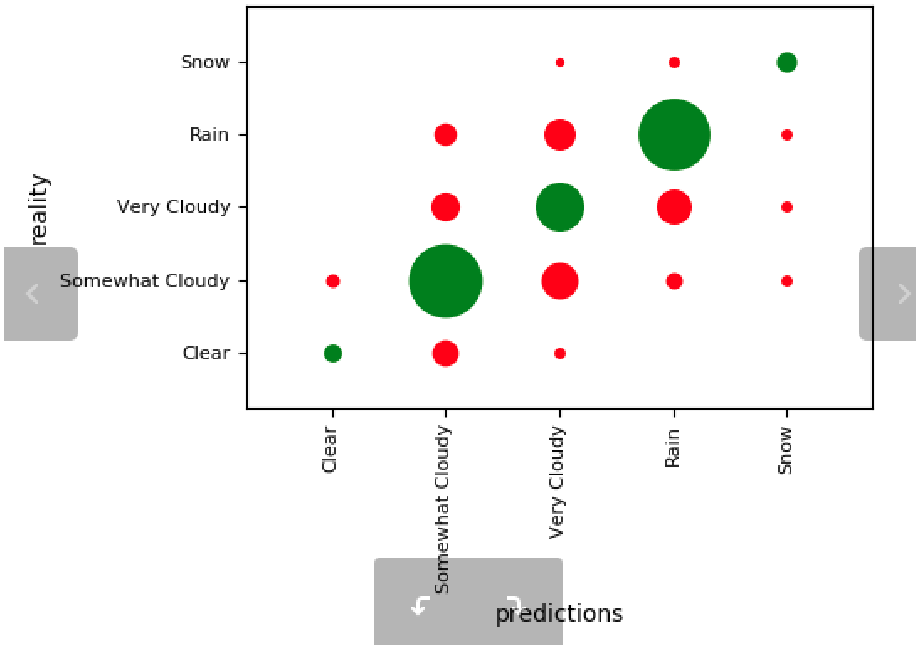
I get an average result of 66% accuracy for my 5 categories. This definitely means that my model is doing something, but whether it is a good result is another story. Because the pictures are all taken from the same spot just hours apart I am afraid that many of the training images may have been too similar to the test images. Then again the lighting in the sky can change a lot in just an hour. Whether it's a good result is hard to say, but looking at the images and predictions, I think it does a decent job.

The classifier worked surprisingly well for snow despite a fairly small sample size. This makes sense though because snow sticks to the ground and changes the whole scene. The classifier does decently to decide if it's raining too. sometimes the lense is wet/blurry, and the clouds are dark, so it's easy to tell why. The big problem is clear weather vs somewhat cloudy weather. My classifier just cannot figure this out and just opts to almost always choose somewhat cloudy since it is far more common. This lack of distinction is likely caused by the image information being based in a different location than the weather information. If it's not very cloudy it's easy to think that one place might be clear while the other cloudy.

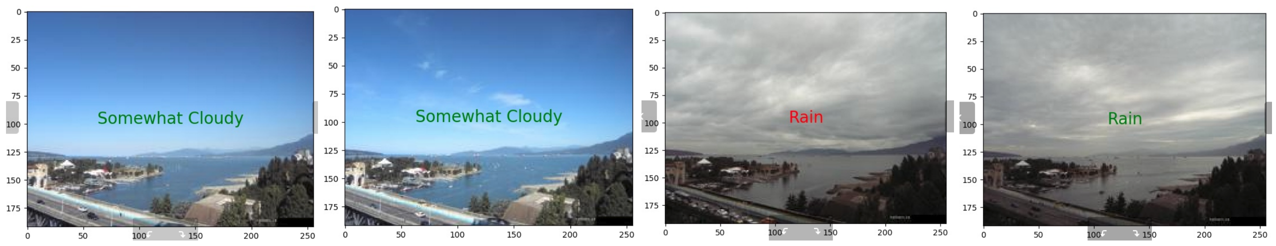
K nearest neighbours did surprisingly very well with only the average brightness and rgb of a pixel. This means that I could have scaled each image to 1x1 and it would have gotten 60percent (actually that's effectively exactly what I did.) I found this very surprising. The first paragraph in this results section may explain this, since some of the images are very very similar. It's notable that I could get this 60percent rating even with k = 1 neighbour.

It's worth nothing that if I combine the categories "Somewhat Cloudy" and "Very Cloudy" I get 80%

5 Visualizations



Green are "correct" predictions, while red are "incorrect"



6 Limitations

Here are some of the major problems:

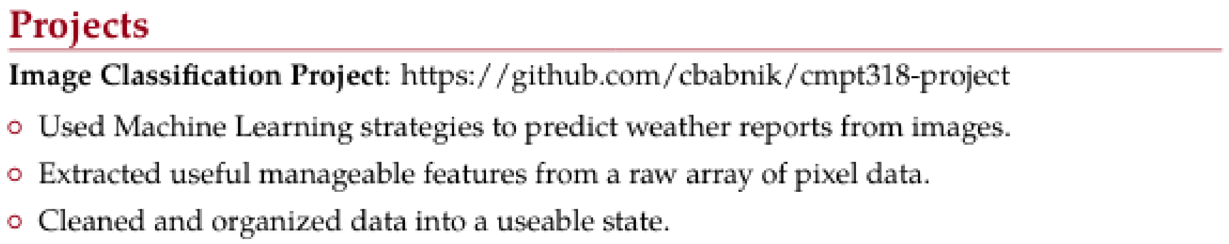
- I didn't address that the locations of weather and images do not match. This is actually a big deal. Consider the mistakes between clear weather and somewhat cloudy. I looked at several pictures that are definitely clear but the weather report says cloudy. English bay could easily be clear while YVR is cloudy and vise versa. This is the greatest difficulty that I have trouble thinking of a solution to.
- I run PCA on all features, but I would rather run PCA only on my pixels, and leave some features such as the overall hue and brightness alone. Perhaps this isn't actually a big deal, but I feel it would be naive to assume that PCA will definitely see how the overall hue and brightness are way more important than any one other pixel feature.

If I spent more time on the project I would look into solving the above, but also:

- Try my classifier for categories like fog vs no-fog
- Try many other ways to represent the image (contours, etc)
- Consider doing something about those black nighttime pics. exclude?
- Try other models like a neural net.

7 Project Experience Summary

In my resume I would include this experience something like this:



Projects

Image Classification Project: https://github.com/cbabnik/cmpt318-project

- Used Machine Learning strategies to predict weather reports from images.
- Extracted useful manageable features from a raw array of pixel data.
- Cleaned and organized data into a useable state.