CMPT​ ​318​ ​Report

# --Topic:​ ​Webcams,​ ​Predictions,​ ​and​ ​Weather

Group​ ​member:

Yifan​ ​Liu

Haosen​ ​Cheng

Qifan​ ​Wu

# Introduction

In this project, our group choose one interesting topic: Webcams, Predictions, and Weather. The reason that we pick this topic is nowadays, computer can easily scan the weather picture and then make a judgement about which weather it is. With each pixel’s RGB value, computer can predict weather quality, which is clear or not clear. By some of the statistic analysis model, we can know what the weather is going to be in the future. It really helps people in the daily life and for the weather​ ​forecast, ​​it​ ​is​ ​even​ ​more​ ​useful.

In this topic, we basically convert each weather picture into pixel table, then we do analysis by statistical model. At last, we use these data to get​ ​the​ ​best​ ​result​ ​for​ ​audiences.

# Refine​ ​data​ ​procedure

## Combine​ ​datasets

Data comes with multiple csv files of readings from weather station and about seven thousand photos. We first combined all csv files to have a general look. We observed that many readings like wind chill and humidity​ ​are​ ​empty,​ ​we​ ​dropped​ ​them.

Also, not all rows are with a weather assigned so we copied what’s next to fill them in. Which means if a row of readings is without a weather assigned, we copy the previous weather to this row. We thought about dropping all rows without a weather, but turns out more than half of the dataset​ ​was​ ​deleted​ ​and​ ​leaves​ ​not​ ​enough​ ​data​ ​to​ ​train​ ​the​ ​model.

​Lastly,​ ​we​ ​removed​ ​rows​ ​with​ ​any​ ​entry​ ​empty​ ​to​ ​finalize​ ​our​ ​dataset.

## Process​ ​images

Images are in size of 192x256 and are RGB, when we read in images, every image becomes a 3d-array of size 192x256x3. For later use in the model we defined to predict weather, we resized every image to size

224x224​ ​to​ ​fit​ ​the​ ​model.

Combine these images with their is not hard, pandas has well defined methods to do that. However, saving them to a file took us a while to figure out how. Because the images are converted to arrays when they are read in, the arrays are too large for encodings to convert them to a file. We first thought csv file can do the job, turns out the saved file can be read by the program but unable to be viewed by a human. To solve that, we changed our data file format from csv to pickle so that the data is stored in binary form without any encoding. However, file is much larger​ ​in​ ​this​ ​format.

## Combine​ ​readings​ ​and​ ​images

We use the dates from both files generated above to create a merged dataset with images and readings corresponding to each other. This, however, left may rows without an image because photos taken are not as​ ​many​ ​as​ ​readings.

We then remove the rows without a image, this leaves the dataset with about 7000 rows from training and testing. At this moment, all data preparation​ ​are​ ​done​ ​and​ ​we​ ​are​ ​ready​ ​to​ ​train​ ​some​ ​models.

# Algorithms

For the project, we choose to use MLPClassifier(the Multi-Layer Perceptron Classifier model) which can build a neural network. The reason we choose this model is it can learn a non-linear model. We found out that Sklearn doesn’t support taking n-dimensional arrays as a training input, which means we cannot create a model that take both the original weather csv and those images taken from the weather station.

So,​ ​for​ ​MLPClassifier,​ ​we​ ​didn’t​ ​use​ ​images​ ​to​ ​train​ ​the​ ​model.

The refined csv we get from the original csv has size of 13156\*11 which we think is a big data. So, the first thing comes to our mind is to use a scaler to reduce the dimension of the input data. Then, the hardest​ ​part​ ​is​ ​to​ ​test​ ​the​ ​parameters​ ​and​ ​get​ ​the​ ​highest​ ​score.

The scaler we use is StandardScaler. The reason is those training inputs (such as temperature and wind speed) should be considered by taking mean/average value. Taking minimum and maximum values as training​ ​inputs​ ​does​ ​not​ ​match​ ​reality.

We choose to use the following parameters: solver, alpha, hidden\_layer\_sizes, random\_state, activation, batch\_size, max\_iter, learning\_rate,​ ​and​ ​tol.

First, we will introduce the non-numeric parameters. For solver, we choose to use ‘lbfgs’ because it can converge faster and perform better. For activation, we choose to use ‘logistic’ which applies logistic sigmoid function and returns f(x) = 1 / (1+exp (-x)). For learning\_rate, we choose to use ‘adaptive’ because it can keep the learning rate constant to

‘learning\_rat\_init’​ ​as​ ​long​ ​as​ ​training​ ​loss​ ​keeps​ ​decreasing.​ ​[1]

Second, we will introduce numeric parameters. Our first idea is to use a nested loop to test each parameter and get the best score. However, it is not possible to testing in this way because we tried to use one loop to test for only one parameters. The iteration time is about 1000 times. It takes a computer more than 20 hour to finish that loop which surprising us.​ ​We​ ​couldn’t​ ​image​ ​how​ ​long​ ​it​ ​will​ ​take​ ​to​ ​finish​ ​6​ ​for​ ​loops.

**The**​​**solution**​​**is**​​**the**​​**following:**

**Step**​​**1:**​ ​choose​ ​a​ ​parameter​ ​and​ ​check​ ​the​ ​score​ ​with​ ​default​ ​value.

Eg.​ ​alpha=0.0001

**Step 2:** test that parameter with two values, one is larger than default

(right​ ​side),​ ​one​ ​is​ ​smaller​ ​than​ ​default​ ​(left​ ​side).

Eg.​ ​alpha=0.001​ ​&​ ​alpha=0.00001

**Step**​​**3:**​ ​test​ ​more​ ​values​ ​on​ ​the​ ​side​ ​that​ ​has​ ​better​ ​score.

Eg.​ ​Larger​ ​alpha​ ​is​ ​better,​ ​so​ test​ ​​more​ ​values​ ​on​ ​right​ ​side.

**Step 4:** For further testing, I will use a number that is much larger/smaller​ ​than​ ​the​ ​default​ ​depending​ ​on​ ​step​ ​3.

Eg.​ ​alpha​ ​=​ ​1

**Step 5:** If the score we get from setp.4 is better, repeat step 4 with a larger/smaller​ ​value.

Eg.​ ​alpha​ ​=​ ​2

Else,​ ​decrease​ ​the​ ​changes​ ​by​ ​50%,

Eg.​ ​alpha​ ​=​ ​0.5

**Step**​​**6:**​ ​repeat​ ​step​ ​5​ ​until​ ​we​ ​get​ ​the​ ​best​ ​score.

Eg. the sequence for testing alpha is {0.0001, 0.001, 1, 2, 4, 3, 2.5} and​ ​the​ ​best​ ​score​ ​we​ ​can​ ​get​ ​is​ ​when​ ​alpha​ ​=​ ​2.

Then the main idea is binary search. And it is much efficient than using loops because the score may stay at same for many different input parameters. We do the above steps for every parameter. By using loop, it is not possible to get the best score in three days. However, we get the best​ ​score​ ​manually​ ​in​ ​three​ ​days.

After the scikit-learn model, we decide to produce something more flexible. We decide to use tensorflow to create two model, one for dataset without images and one for images. We learned tensorflow from scratch​ ​on​ ​Youtube[2].

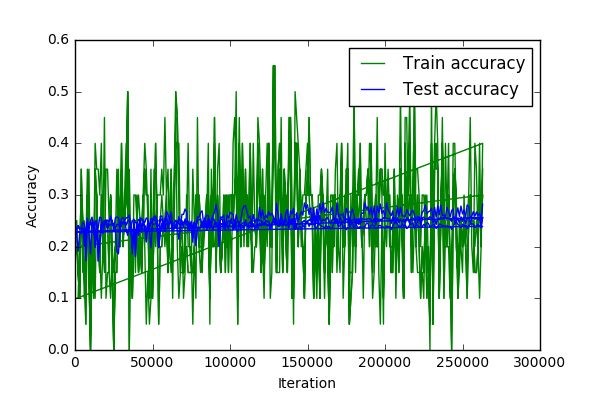
The model we used without images is a full connections neural network with 4 layers, one input layer, one output layer and 2 hidden layer. two hidden layer have 28 and 128 neurals respectfully. Comparing to scikit-learn, this model have more flexibility, it can set learning rate to change while the model is training and have activation function differently​ ​on​ ​each​ ​layer.

Lastly, the model with image included. This model is created by Alex Krizhevsky (google)[3], the picture resize in the data preparation step is for this because alexnet uses images with size 224x224. We think this model is created for some other purpose hence this does not give a promising result as expected, we might change model or implement one ourselves as a side project afterward and for now, we’ll stick to alexnet model.

# Visualization

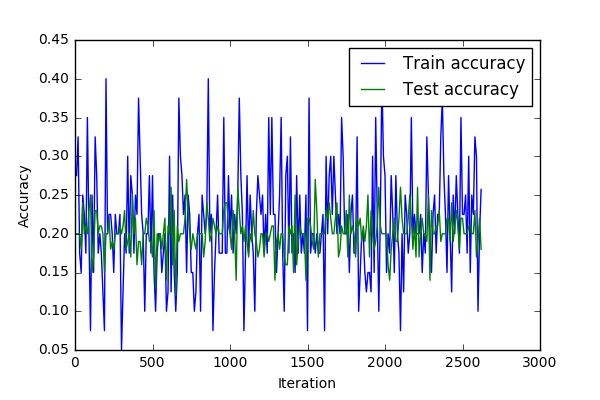
Using tensorflow without image, we got the following accuracy plot. It

doesn’t​ ​gives​ ​a​ ​very​ ​promising​ ​output



**figer**​​**4.1.1**​​**tensorflow**​​**without**​​**image**

and​ ​the​ ​following​ ​plot​ ​is​ ​from​ ​alexnet​ ​model



**figer**​​**4.1.2**​​**tensorflow**​​**with**​​**image**

# Limitations

## Dynamic​ ​pictures

One of the limitations of our design is it only supports static pictures. But in reality, the weather is changing all the time. If we can identify the weather​ ​by​ ​a​ ​piece​ ​of​ ​video,​ ​it​ ​would​ ​be​ ​much​ ​better.

## Weather​ ​predictions

In our project, we can only identify which weather it is and show people the weather conditions. But what if we can make weather predictions by some statistic model and other extra stations information?

I believe with more information, we can make a simple and short weather​ ​predictions​ ​with​ ​more​ ​than​ ​80%​ ​accuracy​ ​rate.

## Accuracy​ ​improvement

At present, the best score that we can get is 86.9%. However, it is not perfect enough. We can either write a simple loop to improve it or change another model to make it better. The problem is if we write a loop, but it takes too much time to execute. And we tried to use multiple models,​ ​unfortunately,​ ​so​ ​many​ ​of​ ​them​ ​are​ ​failed.

## Time​ ​complexity

Until now, the total execution time is still a huge problem. Generally, it takes us hours to test a piece to code, especially in statistic model. The main​ ​reason​ ​is​ ​if​ ​we​ ​train​ ​the​ ​data,​ ​the​ ​algorithm​ ​is​ ​too​ ​much​ ​complex.

**Project** ​ **Experience**​ ​ **Summaries**​

Haosen​ ​Cheng:

Webcams,​ ​Predictions,​ ​and​ ​Weather​ –​ ​CMPT318(Big​ ​Data​ ​analysis)

Sept​ –​ ​Dec​ ​2017

* Collaborate​ ​with​ ​two​ ​other​ ​students ​on​​ ​refine​ ​weather-related​ ​data set​ ​from​ ​GHCN,​ ​develop​ ​machine​ ​learning​ ​models​ ​and​ ​analyze results.
* Create​ ​a​ ​neural​ ​network​ ​machine​ ​learning​ ​model(MLPClassifier) and​ ​improve​ ​the​ ​model​ ​score​ ​(accuracy)​ ​from​ ​40%​ ​to​ ​86.9%.
* Using​ ​Pipeline​ ​to​ ​create​ ​a​ ​model ​that​​ ​first​ ​use​ ​StandardScaler ​​to decrease​ ​the​ ​dimension​ ​of​ ​the​ ​training​ ​data​ ​set​ ​and​ ​then​ ​trained​ ​by MLPClassifier.​ ​This​ ​method​ ​improves​ ​the​ ​efficiency,​ ​it​ ​reduces​ ​run time​ ​from​ ​30​ ​minutes/time​ ​to​ ​15​ ​minutes/time.
* Manually​ ​testing​ ​for​ ​parameters​ ​instead​ of​ ​​using​ ​nine​ ​nested​ ​for loops.​ ​Reduces​ ​calculations​ ​from​ ​millions​ ​of​ ​times​ ​to​ ​less​ ​than​ ​a hundred​ ​times.
* ​ Find​ ​parameters​ ​with​ ​highest​ ​score ​​manually​ ​by​ ​using​ ​binary search​ ​algorithm.

Qifan​ ​Wu:

Webcams,​ ​Predictions,​ ​and​ ​Weather​ –​ ​CMPT318(Big​ ​Data​ ​analysis)

Sept​ –​ ​Dec​ ​2017

1. Data​ ​cleaning​ ​and​ ​preparation,​ ​combine ​datasets​​ ​to​ ​one​ ​for training​ ​a​ ​machine​ ​learning​ ​model.
2. Data​ ​refining,​ ​identifying​ ​and​ ​dropping ​useless​​ ​columns​ ​and​ ​null columns.
3. Report​ ​writing,​ ​changing​ ​layout.
4. Problem​ ​analysis,​ ​find​ ​limitations​ ​and​ ​program​ ​testing.

Yifan​ ​Liu:

Webcams,​ ​Predictions,​ ​and​ ​Weather​ –​ ​CMPT318(Big​ ​Data​ ​analysis)

Sept​ –​ ​Dec​ ​2017

* Algorithm​ ​designer​ ​for​ ​the​ ​project, ​implement​​ ​deep​ ​learning​ ​neural network​ ​with​ ​tensorflow.
* Developed​ ​a​ ​model​ ​and​ ​program​ to​ ​​predict​ ​weather​ ​using​ ​given readings​ ​from​ ​weather​ ​station​ ​and​ ​images​ ​of​ ​English​ ​bay.
* Specialize​ ​in​ ​how​ ​to​ ​make ​precision​​ ​predict​ ​by​ ​modifying parameters​ ​of​ ​tensorflow​ ​model.

Reference

[1]

http://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.​ MLPClassifier.html

[2]

https://www.youtube.com/playlist?list=PLjSwXXbVlK6IHzhLOMpwHHLjY mINRstrk

[3]​https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib /slim/python/slim/nets