Math 480 Final paper

Brian Sherrill

May 12, 2014

Project Description

GITHUB TO FILES: https://github.com/bsherrill480/final

This project is focused on basic image recognition using python. I chose this project to improve upon my understanding of python programming and learn some techniques of image recogition.

For the start of my project I will attempt to classify capital Roman letters. If I have time I would like to move onto other objects.

I have found this project to be much harder than I had expected. The programming and the literature of image recognition has proven to be much more difficult that I expected.

In my project I use Python 2.7 and two packages Numpy and OpenCV. Numpy allows for more efficient operations of data structures such a matrices, and OpenCV is a libray for image processing and manipulation.

Contents

1 Difference method

The method was inteded to be a conceptually and programmingly simple. I did not think this method would be very accurate, but a good place to start. Unfortunatley the programming turned out to be much more difficult than I had expected.

1.1 Conceptualization

The idea was to have a training set of data, and normalize the training set. Then normalize the unknown image via the same process. Then one could find the error (difference between letter in the training set) for each letter in the training set. Ideally the unknown letter, when subtracted from the training set, would leave nothing but zeros and thus would have a small error. To find the correct letter just find the error of the unknown letter with each letter A,B,C,...,Z. The smallest errors represents the true letter.

1.2 Implementation

See https://github.com/bsherrill480/final/blob/master/difference_method.py

1.3 results

To see training data at https://github.com/bsherrill480/final/tree/master/LETTERS and hand-drawn letters at https://github.com/bsherrill480/final/tree/master/HandLetters

TODO: insert training letter and hand drawn letter. better present results

In a small test I got these results. The array of numbers represents the sum of "error" between the training data (in alphabetical order) and the unclassified hand-drawn letter. The second line tells what the hand-drawn letter was and then tells what it was matched.

```
[9321, 17449, 13327, 16850, 15071, 11767, 12370, 11688, 29059, 10499,
13383, 11665, 16730, 15120, 14810, 13222, 15955, 13592, 12335, 9570,
13856, 11089, 13616, 10635, 9518, 11771]
A was matched to:  A
[7218, 5854, 3154, 5836, 5112, 2737, 3320, 3081, 18979, 6920, 5172,
2780, 8597, 6370, 5432, 3444, 6502, 2214, 2326, 7382, 5319, 7703, 9252,
6900, 6350, 5391]
B was matched to:  R
[13945, 16898, 5719, 11523, 12048, 9816, 8119, 14127, 32266, 11855,
12958, 4763, 17265, 15388, 7985, 11338, 9204, 13956, 10681, 10497,
9800, 12408, 14982, 13736, 11668, 12027]
C was matched to:  L
[13825, 9550, 3865, 2017, 9101, 7593, 5299, 8583, 26212, 6855, 9891,
3389, 10514, 9755, 3538, 6484, 4535, 8510, 8203, 8353, 3905, 9952,
10876, 12771, 9863, 10246]
D was matched to:  D
[9312, 9086, 5346, 8505, 3117, 2039, 5933, 6172, 20896, 9259, 6617,
1834, 11499, 9775, 7503, 4379, 8376, 6596, 5905, 6049, 6143, 7921,
9576, 8234, 7318, 6607]
```

```
E was matched to:  L
[11369, 9578, 8071, 8621, 5455, 1680, 9009, 5160, 25045, 11116, 7827,
5364, 11629, 9509, 9516, 2149, 11362, 5518, 9097, 7046, 7788, 9319,
10813, 10662, 8251, 9121]
F was matched to:  F
[10341, 13515, 4941, 10804, 11298, 10304, 5310, 11338, 28800, 8150,
11728, 5716, 14236, 11741, 6731, 11979, 7929, 11904, 8228, 10328, 8327,
12093, 13744, 12321, 11566, 11773]
G was matched to:  C
```

1.4 room for improvment

I ran into a few problems:

First was an issue of normalization. I would like a block average but I'm using OpenCV's resize feature which may be cutting out details to make it look more presentable.

Second, I ran into an issue of scaling. To cut out white space, I measued the quantity of black in each row/column and cut out stuff before and after a certain "tollerance" (when each column/row contained a certain percentage of the total black in the image). I think this may be probelmatic because as the image got larger, the amount of total black per row/column becomes less so the tollerance doesn't scale with size. To fixed this I rescaled the image but I feel there should be a better way.

Third, I know there may be a better way to find error than by just subtracting one image from another.

Fourth, I used only a single image for the training data, it could probably be improved by using many images for the training set.

2 TODO: k-means

2.1 Conceptualization

Use a large sample size as trainig data and form clusters.

2.2 Implementation

2.3 results

2.4 Improvments

3 TODO: 1D FFT to extract frequiencies and compare to kmeans

TO DO

3.1 Conceptualization

3.2 Implementation

3.3 results

3.4 Improvments

4 TODO: 2D FFT to extract frequiencies and compare to kmeans

TO DO

4.1 Conceptualization

4.2 Implementation

4.3 results

4.4 Improvments

5 TODO: PCA to extract pattern

5.1 Conceptualization

5.2 Implementation

5.3 results

5.4 Improvments