ECE532 Final Project Update 1

Timothy Winfree

November 2020

1 Pre Processing/Feature Extraction

One approach to this problem is to do no pre processing and simply unwrap each column of each image and stack them into a feature vector. Since every digit of the same class is drawn differently, the relationship between the pixel values and the correct label will be weak. To provide a more meaningful set of features, I want to extract structures from the images, whos existence and location in the image have a strong connection to a subset of the classes. To accomplish this I convolved each image with the pattern matching filters. The output of each of these filters is subsampled to 16x16, then unwrapped and concatenated into a feature vector. A feature vector is produced in this manner for each image and then stored as a row in a matrix X.

The purpose of subsampling is that the precise location of a particular structure is not important because every digit is drawn slightly differently, with its structures in different places. As such, I just want to know the general location were the template matched the image. For example, If I know that I have a horizontal line in the upper part of the image, and I slanted line in the middle, I have a seven.

In addition to pattern matching, I implemented a Harris corner detection function, which takes an image as an input, and outputs the location of corners/junctions in the image (if there are any). I also implemented an experimental dilation-erosion function, whose purpose is to identify the existence/location of closed loops. (See fig for demonstration)

2 Implementation of Linear Regression for multi class learning

Initially I tried training a single set of classifier weights to segregate all 10 classes. With missclassification rates of 75%, it became evident that least squares is not a good technique for solving non binary classification problems. Instead I divided this non binary problem into a series of 9 binary "i or else" classifications with i being the labels 0-8. (My implementation of this algorithm and its supplementary functions can be found at the project link). In practice I compute a unique set of classifier weights w_i for each of the 9 cases. Prediction of the rth image will at the ith node in the decision tree takes place as follows: If the array sign(X(r,:) * Wi) contains +1, then i will be the prediction, else you continue to the (i+1 or (i+2)-9) node. At node i = 8, if the label is -1, 9 is the prediction.

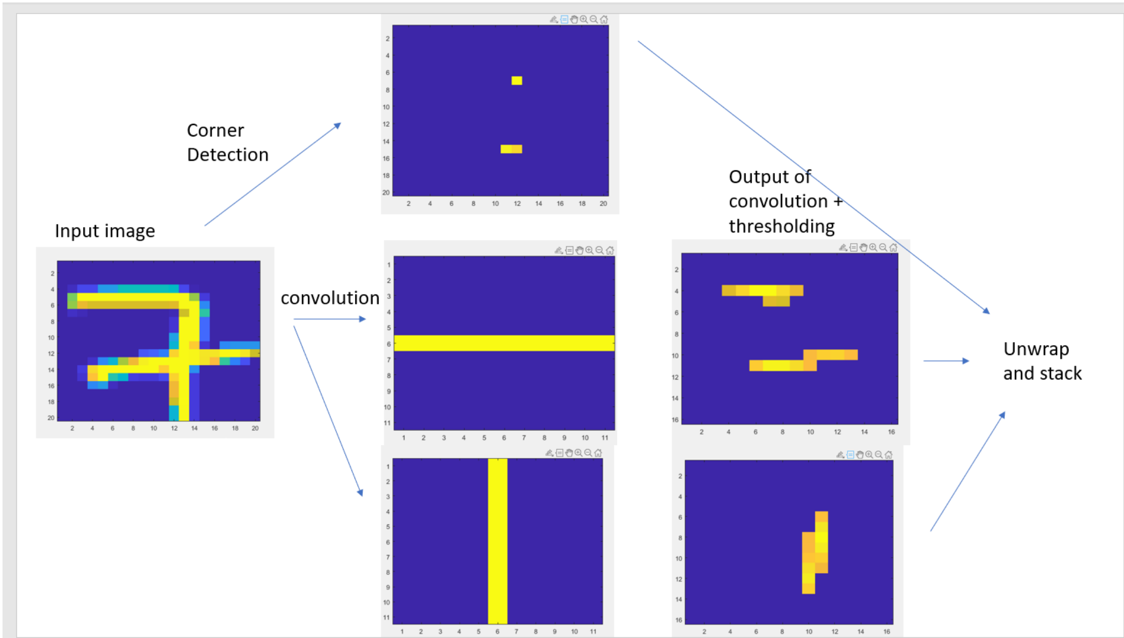

Figure 1: Example of pattern/template matching with convolution and corner detection

3 Next Steps

The next steps are to use my implemented binary regression tree to analyze different methods of regularizing the least squares problem. I will use cross validation to tune regularization parameters independently for each of the 9 binary classifiers, and compare the results of regularizing via Tikhonov, lasso, and truncated SVD regression. Beyond this, I will complete the implementation of the KNN function and compare its performance with linear regression.

4 Updated Timeline

The preprocessing stage was more involved than I anticipated. As such, I have to push back my analysis of linear regression and k nearest neighbors to the second update, Dec 1. The convolutional neural network and synthesis of the final report will be done by the final deadline, Dec 12.

4.1 Link to Project

`https://github.com/twinfree/ECE532-Final-Project`

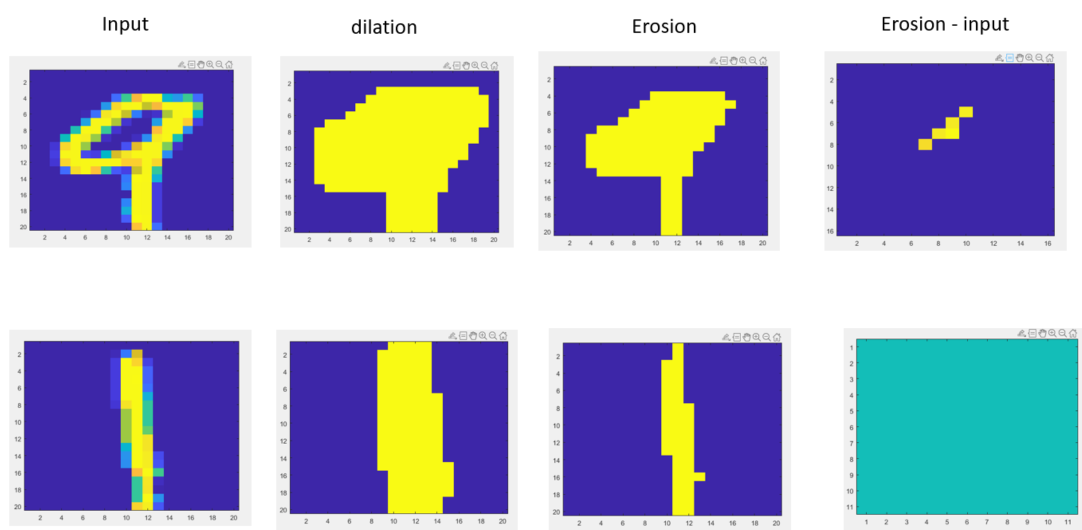

Figure 2: Example of dilation-erosion method. The initial dilation closes any loops, which is not undone by the subsequent erosion. Thus, when you take the difference between the dilate-eroded image and the input, their will be a large value at the center of the closed loop.