Tiffany Chen

ID: 998840686

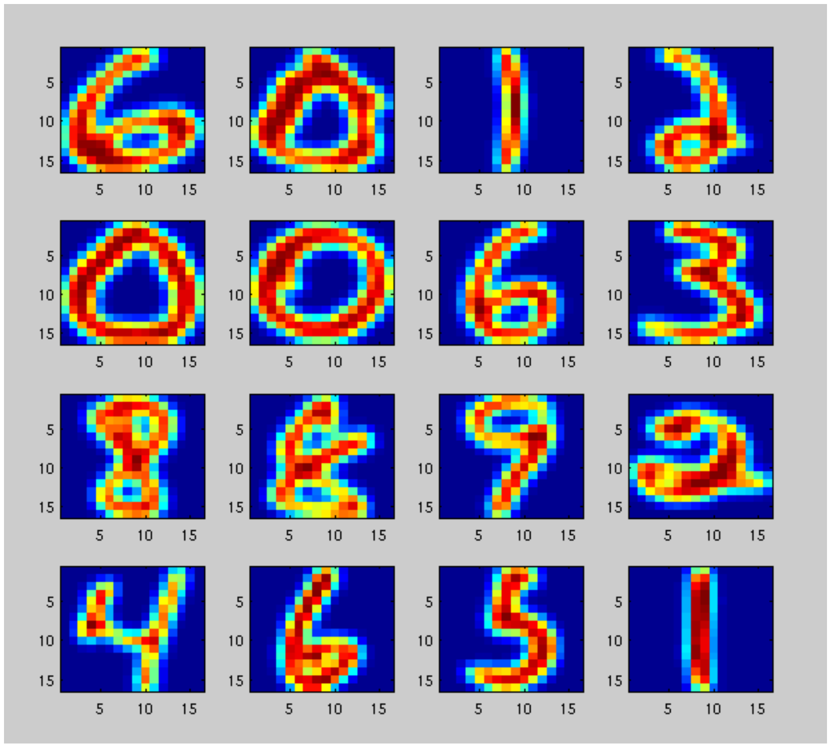
5/27/16

MAT 167 Digits Recognition Project

Introduction

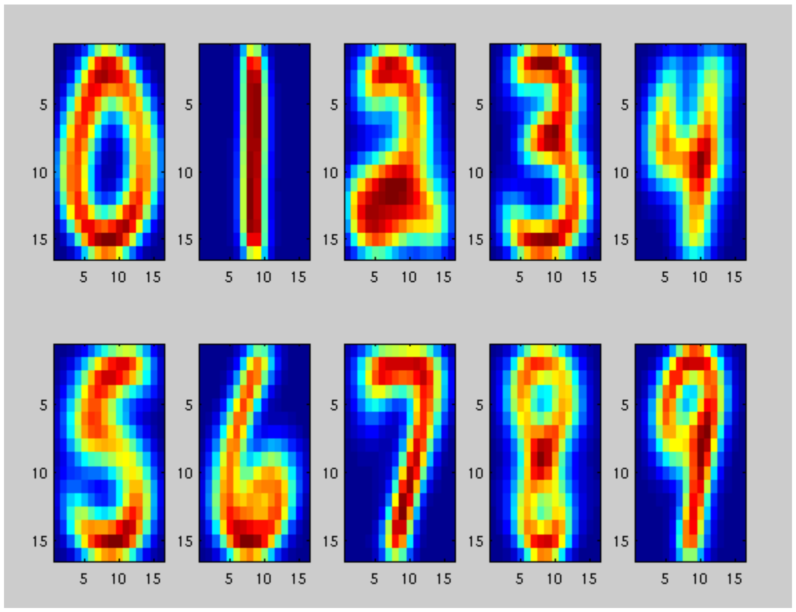
Classifying handwritten digits by computer programs is a classic problem in pattern recognition. For this project, we take a look at the USPS digits data set offered free online and from the CEDAR (Center of Excellence in Document Analysis and Recognition) in New York Buffalo SUNY. This write up is broken up into 5 steps and was completed in Matlab. The first step is taking a look at the first 16 images in the data set and displaying it. The second step is to display the average of each digits. Step 3 is to use the simplest classification computation, Euclidean distance to recognize the digits, with a confusion matrix that follows. Step 4 is similar to step 3 but using the SVD based classification computation. Lastly, step 5 is a summary of all of the above, with detailed explanation of the data and theories. We compare and contrast the two classification algorithms and notice that SVD is better and more accurate.

Step 1



The figure above displays the first 16 images in train patterns.

Step 2



The figure above displays the 10 mean digit images. In other words, for each digit, the one digit represented above is the average of all the digits for that ”kind”.

Step 3
Below displays the test confusion matrix for using the simple classification algorithm with only Euclidean distance being considered. The rows represent the actual labels and the columns represent the predicted labels. Thus, we would want the diagonals to have the most values because it indicates accuracy.

$$\text{test confusion} = \begin{pmatrix} 656 & 1 & 3 & 4 & 10 & 19 & 73 & 2 & 17 & 1 \\ 0 & 644 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 14 & 4 & 362 & 13 & 25 & 5 & 4 & 9 & 18 & 0 \\ 1 & 3 & 4 & 368 & 1 & 17 & 0 & 3 & 14 & 7 \\ 3 & 16 & 6 & 0 & 363 & 1 & 8 & 1 & 5 & 40 \\ 13 & 3 & 3 & 20 & 14 & 271 & 9 & 0 & 16 & 6 \\ 23 & 11 & 13 & 0 & 9 & 3 & 354 & 0 & 1 & 0 \\ 0 & 5 & 1 & 0 & 7 & 1 & 0 & 351 & 3 & 34 \\ 9 & 19 & 5 & 12 & 6 & 6 & 0 & 1 & 253 & 20 \\ 1 & 15 & 0 & 1 & 39 & 2 & 0 & 24 & 3 & 314 \end{pmatrix}$$

Step 4
Below displays the test confusion matrix for using SVD based classification algorithm. In this case we use the rank 17 SVD of the image sets. Again, the rows represent actual labels and columns are the predicted labels. Right away by glancing at the two matrices, this SVD one has less error (or values) in the off diagonals which indicates that SVD algorithm is more accurate than using simple.

$$\text{test svd17 confusion} = \begin{pmatrix} 772 & 2 & 1 & 3 & 1 & 1 & 2 & 1 & 3 & 0 \\ 0 & 646 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 3 & 6 & 431 & 6 & 0 & 3 & 1 & 2 & 2 & 0 \\ 1 & 1 & 4 & 401 & 0 & 7 & 0 & 0 & 4 & 0 \\ 2 & 8 & 1 & 0 & 424 & 1 & 1 & 5 & 0 & 1 \\ 2 & 0 & 0 & 5 & 2 & 335 & 7 & 1 & 1 & 2 \\ 6 & 4 & 0 & 0 & 2 & 3 & 399 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 2 & 0 & 0 & 387 & 0 & 11 \\ 2 & 9 & 1 & 5 & 1 & 1 & 0 & 0 & 309 & 3 \\ 0 & 5 & 0 & 1 & 0 & 0 & 0 & 4 & 1 & 388 \end{pmatrix}$$

Step 5

Part a
Given a set of manually classified digits (training set), we want to classify a set of unknown digits (test set). So in this dataset USPS, there are a certain number of digits 9398, relatively equally distributed between 0 to 9 and the test set also has half 4649 and the training has the other half 4649. A little background on the USPS dataset is that was gathered at the Center of Excellence in Doc Analysis and Recognition at SUNY buffalo, sponsored by the USPS. Train labels and Test labels is a 10x4649 matrix where the columns represent a certain digit from 0-9.Where the 1 is located in the column, corresponds to what the digit actually is. The rest of the entries in the column is -1. More specifically, the arrays contain the true information about the digit images. Additionally, the jth handwritten digit image in train patterns truly represents the digit i and the (i+1, j)th entry of train labels is 1, while the other entries of the jth column is -1. The train patterns and test patterns are matrices of 256x4649 that contains numbers that represent a raster scan of the 16x16 gray level pixel intensities that have been normalized to lie with the range [-1,1]. That is the basic background on the data set.

Part b
In step 3, we want to classify the unknown digits by using Euclidean distance. This means, in the training set, given the manually classified set of digits, we compute the means of all the 10 classes/kinds of digits which is what we do in step 2. How we did it was we pooled together all the digits of one "kind", then we averaged them. Then, for each digit in the test set, classify it as that digit if it is the closest mean (or closest in Euclidean distance). If we consider the training set digits as vectors or points, then we can assume the digits of one kind form clusters in a Euclidean space. These clusters are well separated if the training set digits are well written. From the output, the digits are mostly well written because we illustrate the means of the digits so the clusters are well separated, and it is easy to recognize the digit.

Part c
Overall the SVD algorithm is better than using the squared Euclidean distances. This is because the values in the diagonals are greater than in the Euclidean case. Also, by calculating the accuracy percentage of each digit and putting it into a matrix below, we can determine how each digit performs relative to the two algorithms. The percentage is calculated by dividing each entry of the confusion matrix by the sum of the rows since it represents the actual labels. From these matrices, the overall Euclidean classification rate is 84.66 percent and for SVD, it is 96.225 percent, which also shows the classification rates were considerably better than the simple classification algorithm. Also, the digit most difficult to identify correctly is 5 for euclidean and 8 for SVD. While the digit easiest to identify correctly is 2 for SVD and euclidean. These make sense, because 5 and 8s tend to have odd loops that people can easily write differently, whereas a 2 is

just one stroke. I'm surprised something like the digit 1 is not the easiest to identify correctly, but maybe this is because some people write it with an extra bit on top which can be mistaken for a 7. Is it difficult to get a 100 percent accuracy because some digits are simply way too hard to discern.

confusion matrix percentage for Euclidean =

$$\begin{pmatrix}
0.8346 & 0.0013 & 0.0038 & 0.0051 & 0.0127 & 0.0242 & 0.0929 & 0.0025 & 0.0216 & 0.0013 \\
0 & 0.9954 & 0 & 0.0015 & 0 & 0 & 0.0015 & 0 & 0.0015 & 0 \\
0.0308 & 0.0088 & 0.7974 & 0.0286 & 0.0551 & 0.0110 & 0.0088 & 0.0198 & 0.0396 & 0 \\
0.0024 & 0.0072 & 0.0096 & 0.8804 & 0.0024 & 0.0407 & 0 & 0.0072 & 0.0335 & 0.0167 \\
0.0068 & 0.0361 & 0.0135 & 0 & 0.8194 & 0.0023 & 0.0181 & 0.0023 & 0.0113 & 0.0903 \\
0.0366 & 0.0085 & 0.0085 & 0.0563 & 0.0394 & 0.7634 & 0.0254 & 0 & 0.0451 & 0.0169 \\
0.0556 & 0.0266 & 0.0314 & 0 & 0.0217 & 0.0072 & 0.8551 & 0 & 0.0024 & 0 \\
0 & 0.0124 & 0.0025 & 0 & 0.0174 & 0.0025 & 0 & 0.8731 & 0.0075 & 0.0846 \\
0.0272 & 0.0574 & 0.0151 & 0.0363 & 0.0181 & 0.0181 & 0 & 0.0030 & 0.7644 & 0.0604 \\
0.0025 & 0.0376 & 0 & 0.0025 & 0.0977 & 0.0050 & 0 & 0.0602 & 0.0075 & 0.7870
\end{pmatrix}$$

confusion matrix percentage for SVD =

$$\begin{pmatrix}
0.9822 & 0.0025 & 0.0013 & 0.0038 & 0.0013 & 0.0013 & 0.0025 & 0.0013 & 0.0038 & 0 \\
0 & 0.9985 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0015 \\
0.0066 & 0.0132 & 0.9493 & 0.0132 & 0 & 0.0066 & 0.0022 & 0.0044 & 0.0044 & 0 \\
0.0024 & 0.0024 & 0.0096 & 0.9593 & 0 & 0.0167 & 0 & 0 & 0.0096 & 0 \\
0.0045 & 0.0181 & 0.0023 & 0 & 0.9571 & 0.0023 & 0.0023 & 0.0113 & 0 & 0.0023 \\
0.0056 & 0 & 0 & 0.0141 & 0.0056 & 0.9437 & 0.0197 & 0.0028 & 0.0028 & 0.0056 \\
0.0145 & 0.0097 & 0 & 0 & 0.0048 & 0.0072 & 0.9638 & 0 & 0 & 0 \\
0 & 0.0050 & 0 & 0 & 0.0050 & 0 & 0 & 0.9627 & 0 & 0.0274 \\
0.0060 & 0.0272 & 0.0030 & 0.0151 & 0.0030 & 0.0030 & 0 & 0 & 0.9335 & 0.0091 \\
0 & 0.0125 & 0 & 0.0025 & 0 & 0 & 0 & 0.0100 & 0.0025 & 0.9724
\end{pmatrix}$$

For SVD, what we do is consider the images as 16x16 matrices. Then the columns of a matrix consisting of all the training digits of one kind will span a linear subspace of R m. To model the variation within the training and test sets, we use orthogonal basis of the subspace. We use the first 17 left singular vectors of the SVD of the training digits. For each digits, we pool the training images corresponding to the digit and compute the SVD. Thus, for the training set of known digits, compute the SVD of each set of digits. And for the classification, for a given test digit, compute the euclidean distance in all 10 bases. If one of them is significantly smaller, then we would classify the digit as that otherwise stop. In the "real" SVD case we would use the residuals from least squares, but that is harder to implement so for the project I just used similar procedures from step 3 and used the euclidean distances. The SVD assumptions are that each digit is well characterized by a few of the first singular images of its own kind and that the expansion coefficients discriminates well between the different classes of digits. Also, if the unknown digit is well classified for one particular basis than another, then its likely the unknown digit is that number. For the training set of known digits, compute the SVD of each set of digits. For more detailed explanation on the theory, see part d.

Part d

All in all, the SVD based classification algorithm yields better results than the simple classification algorithms using Euclidean distance, which we observed by the accuracy of the confusion matrices. The SVD confusion matrix had higher percentage than simple classification, and the overall accuacy was much higher. This is due to using different bases to represent the digits. The simple algorithm doesn't take into account the variation of the digits of one kind, whereas the SVD algorithm does. If A is in R m by n, with m = 256, and is the matrix that consists of all the training digits of one kind, then the columns span a linear subspace of Rm. The SVD will model the variation within the set of training digits of one kind using orthogonal bases of the subspace, which can be computed by SVD and approximated by a sum of rank one matrices for some value k, which we used 17. Each column in A is an image of the one digit so the left singular vectors are an orthogonal basis in the image space of the specific digit. Then, singular vectors will represent the images.

Conclusion
In conclusion, this digits recognition project was a way to apply the theories learned in the classroom to a real world application. I was able to see, after a walkthrough step by step of how to visualize and calculate certain matrices, that the SVD based classification algorithm is better than simple classification algorithms. It was pretty cool. The end!