In this write-up, I (working alone) will first describe my objectives for this project. I will follow this with a discussion on my method for choosing and tuning machine learning models, the results I achieved, and analysis of the data. I will finish this by discussing the ethics of the project.

Objective Description

For this project, I use a variety of classification models to correctly recognize gray-scale digits according to twenty-eight by twenty-eight arrays of pixels.

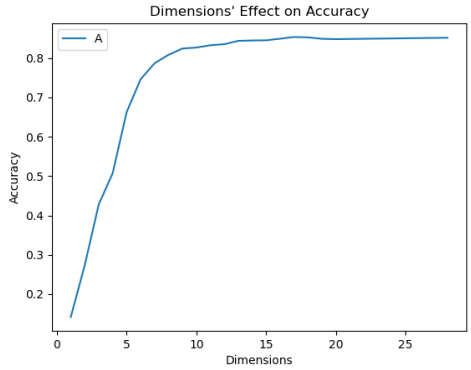
Method Description

I measure the "correctness" or "success" of a model by its accuracy and by its speed – how quickly we can train and predict results. We can use accuracy as a metric because there is roughly an equal distribution among the various digits – see my data exploration. I also argue that accuracy is better than f1 score, since we only care about true positives and true negatives (Huilgol). We want to guess the correct digits, which is what accuracy measures. Conversely, f1 scores considers false negatives, but, in this case, there is no harm to not guessing a specific digit (Huilgol). I am also considering an algorithm's speed, as this is important for real-world uses in applications which use computer vision. Maximizing accuracy and speed, I implemented three different models – decision tree, logistic regression, and k-nearest neighbors. I chose these three models as these were not only the models we implemented in class but also the models I came across in my research of the mnist dataset. I wanted to try all three, seeing which model performs the best.

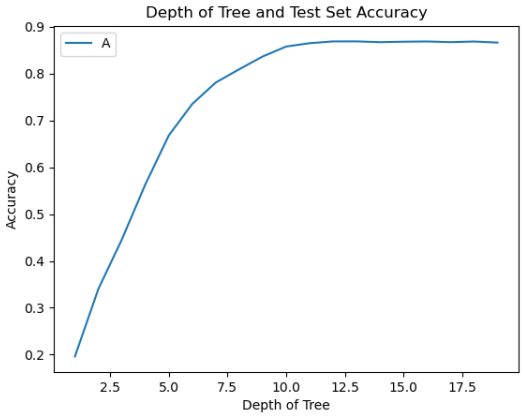
I perform similar steps for pre-processing in each case. I first reshape the input arrays from twenty-eight by twenty-eight matrices to vectors, each containing 784 values between zero and 255. I then scale the input vectors, dividing them by 255. This sort of scaling of the matrices

is possible because the values in the vectors do not exceed 255 and helpful because the computer may quickly calculate *sparse matrices*. That is, many of the pixels in the mnist dataset are set to 0 – i.e., blank. To be sure, the mnist dataset includes 60,000 data points each with 784 values. Given the complexity of the dataset, I consider training the models on a subset of the original data. As such, I hoped to achieve similar results, while training the data on a more manageably-sized dataset. I also consider singular value decomposition to reduce the run-time and hopefully not sacrifice a great deal of accuracy. But these data manipulations differ depending on the model in question. Now, I will go into more depth on my specific models.

a) Decision Tree: for this model, I decided not to perform dimensionality reduction, nor did I reduce the size of the dataset. I did not deem it necessary, for the time spent reducing the dimensions exceeded the time saved when training the model. Nonetheless, the following graph illustrates the effect of dimensionality reduction on the accuracy. To be sure, the accuracy of the graph stabilizes between twelve and 784.



I also consider limiting the depth of the tree to ten. I do so because of the following result:



With a depth of ten, the model achieves the greatest accuracy on the test set.

b) Logistic Regression: much like the decision tree, I did not adjust the training set in any way for the logistic regression. Any dimensionality reduction or subsetting of the dataset would not have saved time but would have potentially affected accuracy. Thus, it would not have been worth changing the dataset. Also, unlike the other models, sklearn's logistic regression does not have hyper parameters to tune. The only adjustment I made was to perform cross validation.

c) K-Nearest Neighbors (KNN): I first implement KNN algorithm and cross validation. This helps to find an optimal k value, and I reason that sklearn's algorithm would find that value faster than my algorithm made from scratch, which is true. I use sklearn's grid search algorithm to find that the optimal k value is twelve. Before using grid search, I used singular value decomposition to reduce the dimensionality to twenty-eight, and the algorithm still took fifteen minutes to run. Finding twelve to be the optimal k value with sklearn, I apply what I learned to the algorithm I built from scratch. Due to time complexity and run time of my algorithm, I include 20,000 examples, one third of the data set. This algorithm took an hour to go over 250 examples, but I thought 250 was a sufficient number of examples.

Results

For each sklearn model, I will show the average accuracy based on cross-validated data and show the corresponding confusion matrix. Note that I do not use a cross-validated model for the confusion matrix.

Decision Tree: using the grid search, the decision tree had an accuracy of .8 (80%).

Cross Validation Scores: [0.808, 0.807, 0.801, 0.790, 0.822]

```
[1157,    0,    6,    1,    2,   13,    6,    5,    4,    0],
[   0, 1250,    4,    4,    2,    5,    0,    1,    5,    1],
[   9,   13, 1076,   18,   11,    7,   11,   16,   29,    6],
[   6,    2,   25, 1103,    1,   58,    7,    7,   24,    9],
[   5,    9,    2,    4, 1121,    1,    8,    2,    5,   30],
[  12,    2,   16,   30,    8,  966,   24,    8,   22,    9],
[   4,    4,   11,    0,   11,   11, 1125,    0,    3,    1],
[   3,    5,   12,   10,    7,    3,    0, 1224,    1,   32],
[   9,   23,    9,   27,    2,   27,    6,    8, 1067,   20],
[   5,   10,    3,   13,   26,    7,    1,   35,    9, 1038]]
```

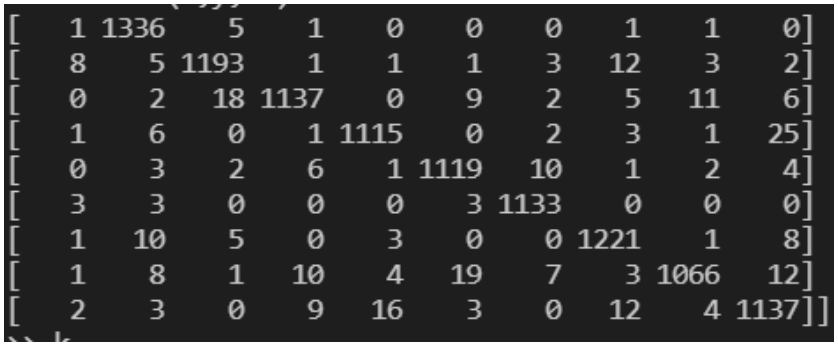
Logistic Regression: using k-fold validation, the logistic regression model had an accuracy of .92.

Cross Validation Scores: [0.919, 0.93, 0.919, 0.917, 0.92, 0.92]

```
[   1 1338    4    0    0    0    0    1    1    0]
[   7    0 1204    1    2    0    2   10    2    1]
[   1    1   13 1144    0    9    0    4   13    5]
[   1    4    0    0 1126    0    2    3    1   17]
[   0    1    2    5    2 1123   10    2    2    1]
[   3    2    0    0    0    2 1135    0    0    0]
[   1    9    3    1    2    0    0 1225    1    7]
[   1    6    4    7    3   14    3    3 1080   10]
[   2    4    0   10   14    3    0   10    2 1141]]
```

K-Nearest Neighbors Sklearn: Using cross validation, we find that the optimal number of neighbors is twelve. The accuracy of this model was .97 (97%)

Cross Validation Scores: [0.981, 0.990, 0.972, 0.975, 0.976, 0.984



]

Cross validation revealed that the best k value was 12, best_param_.

Cross Validation Scores: [0.970, 0.971, 0.969 , 0.968, 0.970]

The accuracy of sklearns was 0.97 (97%)!

Made From Scratch:

My model made from scratch achieved an accuracy of 97%.

Analysis

By my metrics for success, k-nearest neighbors and logistic regression algorithm perform the best. To be sure, KNN is slightly more accurate than logistic regression, but logistic regression predicts results far faster. This saved time could be useful when running computer vision on various applications. Thus, I have determined that these two algorithms are equally as effective. Far behind both models, the decision tree lacks the accuracy of logistic regression, nor does it over perform in terms of speed. Concerning accuracy, we can also look at the confusion matrix to identify interesting trends. In particular, the decision tree and KNN algorithm performed poorly when the result was 3. The algorithms often confused 3 with 9, likely because both numbers contain similar curvatures. In contrast, these three algorithms identified when the true digit was one with great success. I postulate the 1 simply looks different from the other numbers, and it is written in fairly standard ways.

I believe it is also useful to ask: why each model performs the it does? Why do KNN and logistic regression perform well and does the decision tree perform worse? To this, I say that KNN and logistic regression are more accurate because they more holistically consider the curvature in the data. KNN does consider this factor more than the other models, since it always chooses the classifications of the near-by points. Logistic regression, however, is limited by a function which restricts such free curvature. Hence, the model is less accurate than KNN. Moreover, decision trees do not consider curves in the data. Thus, the decision trees do not fit the data as nicely as the other models.

To be sure, I am commenting on what I observe in the data. However, the models I have run have many limitations that we do not see. For instance, suppose we expanded the image and place the digits in random parts of the image – some appear in the top left, others in the bottom right. Our algorithms could not handle such a case. In particular, KNN would have trouble identifying where the digits are supposed to be placed. I suppose then that we could train a stronger model to identify the patterns in the data, one which could detect digits wherever they appear. KNN and logistic regression perform great on small static images of mail-box numbers.

Ethics:

Relative to other experiments in AI and machine learning, the ethics of recognizing digits of mail boxes appears relatively benign. There are far fewer ethical concerns with such digit recognition than with gene editing, health registries, internet searches, and the like. Yet there are still a few ethical concerns with the analysis of the mnist dataset, and the first relates to the collection of the data. The act of finding mailboxes across America is somewhat invasive. Perhaps it was against the will of the owners of the house. Additionally, this dataset may misrepresent those not necessarily from marginalized communities but from rural America. Simply put, it is easier to collect data points from densely populated cities than it is to do the same in the country. This is

a form of inequity in and of itself. Yet, relative to the data collection taking place on each of our smart phones, the collection of mailbox numbers appears relatively benign and less impactful.

Citations:

Huilgol, Purva. "Accuracy vs. F1-Score." The Medium, August 24, 2019 Year;

https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2