Optimizing Image Recognition of Digits

Rachel Carroll

04/17/2020

Abstract

In this paper, a model was created to identify recognize a handwritten digit from an image. First, several mappings from the image data to the identified digit were built using various linear model solvers. These mappings were analyzed to find an optimal model that correctly labeled the images using as few pixels as possible. Accuracy based on number of pixels used was measured and relevant pixel locations were identified for overall performance as well as for individual digit identification. The results showed that using models which promote sparsity by penalizing the L1 norm, was ideal for optimizing the results using few pixels. If too much weight was applied to the L1 norm, the best possible error would remain high as more pixels were added, allowing less sparse solutions to outperform it.

1 Introduction and Overview

For this project, the MNST dataset of 60,000 training and 10,000 testing images of handwritten digits from 0-9 was used to build a model that correctly identifies the digit using the pixel values of the image. The images were vectorized and the pixel data were stored in a matrix. Several models were built using different linear regression solvers to determine a mapping from the image pixel data to the label identifying the digit. The various solvers used different regularizer constraints which effected both pixel importance and accuracy of the model results. These models were analyzed to determine optimal methods to identify the digits correctly using as few pixels as possible. This was performed for overall accuracy of all digits as well as the accuracy of identifying all digits at one time.

2 Theoretical Background

Regression Models for Under/Over Determined Systems

In a typical linear regression model, the overall goal is to solve $AX = B$ where A contains measurements and B contains outcomes of some system of interest. The solution X tells us the coefficients applied to A to arrive at the outcomes in B. In this sense the matrix X acts as a mapping from A space to B space. In other words, it solves the following equation.

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i$$

for $i = 1, 2, \ldots m$ where m is the number of observations/number of rows in A and n is the number of measurements per observations (number of columns of A).

From basic linear algebra principles, if $m = n$ this system will have a unique solution. However in practical, real-world problems, the systems of interest are essentially always going to be either under or overdetermined. Therefore there could be infinitely many solutions (if it's an underdetermined system) or no solutions (if it's overdetermined).

Since we cannot exactly solve $AX = B$, the goal is find the find the "best" solution to the system. The key to model creation is how we define what the "best" solution is. This idea is described in the section below!

Regularization and the L1 vs. L2 norm
Because there are no or infinitely many solutions in over/under determined systems, a constraint or regularization must be defined to allow for a unique solution. As stated the goal of the model is to find the best solution, which is the solution where AX is as close to B as possible. The way we determine closeness comes from the regularization. For example, a very common regularization is the leased square, which yields the solution that minimizes the errors in terms of the $L2$ norm. The L2 is error is as follows

$$E_2 = \|AX - B\|_2 = \left(\frac{1}{m} \sum_{i=1}^{m} |a_i x - b_i|^2 \right)^{\frac{1}{2}}.$$

Another common regularizer minimizes the $L1$ norm which is shown below.

$$E_1 = \|AX - B\|_1 = \frac{1}{m} \sum_{i=1}^{m} |a_i x - b_i|$$

Many different types of regularizers can be made using different types of norms. One of the most prominent attributes of the L2 norm, is that it is highly sensitive to outliers. This is because the differences between the modeled and observed outcomes are squared. So, one large outlier would pull the regression toward itself and away from the majority of the data. Regressions using L1 norm regularizers, on the other hand, are less sensitive to outliers and promote sparsity of coefficients in the solution matrix (X). Because of the sparsity characteristic, promoting the L1 norm is a good way to identify measurements that contribute to the most significant dynamics by eliminating measurements associated with 0 coefficients.

The choice of regularizer has a significant effect on model results. Often, a combination of the L1 and L2 norm are used to create a balance of sparsity and data fitting. In this paper we use many different solvers, which use different regularizers, to identify an optimal model for the given data. The ideal model is dependent on the data and the goal of the analysis.

3 Algorithm Implementation and Development

The following steps were performed in PYTHON

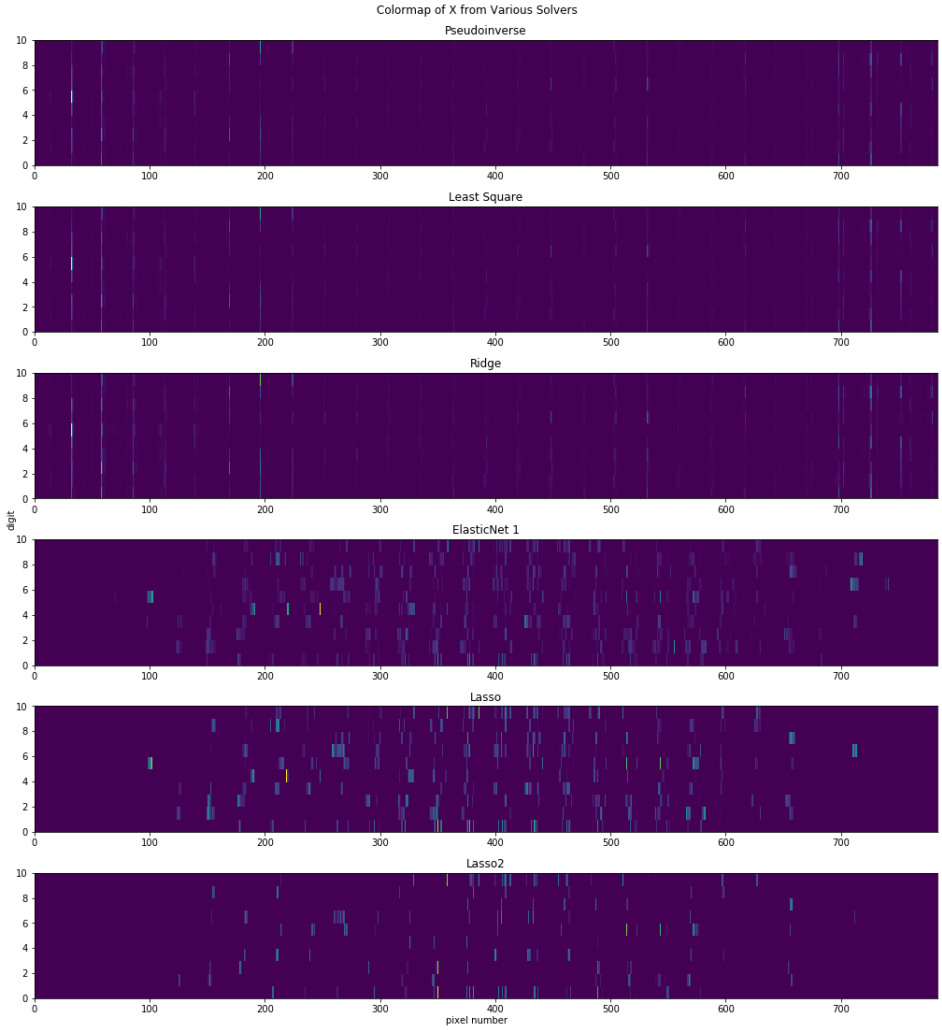
1. Read in images
2. Vectorize each image from a 28×28 matrix to a 1×784 row vector of pixels
3. Store vectorized images in a matrix, A. Here, the ith row of A contains the pixels of the ith image and the $j$the column contains the color value of the jth pixel of the associated image.
4. Create the label Matrix B, where each row is a vector of 0s and one 1 with 10 entries. The location of the 1 indicates what the handwritten digit is. e.g $[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$ means the image is a 2.
5. Use various solvers to solve $AX = B$ for X. This step results in several X matrices that differ depending on the solver used. Here X is a 784×10 matrix which acts as a mapping from pixel to label space. The ith column of X contains pixel coefficients which determine if the image is of the ith digit. For example, if the dot product of a vector of image pixels with the first column of X is very close to 0, then the digit is unlikely to be 1. If however it is high (close to 1) it is likely to be an image of the number 1.
6. For each X, average the columns using the 2 norm to identify the overall significance of each pixel. This allows the most important pixels (according the given solver) to be identified and ranked.

7. Create a model that uses X and a given number of pixels to identify the labels of the images in the training set. The pixels used will be from the ranked list of pixel numbers, from most important to least.
   (a) read in set of images (from training will use matrix A)
   (b) calculate $P = AX$ where P is in the form of B and contains the precited label vectors
   (c) create a list of label numbers based on the position of the highest value in each of the rows of P
8. Run the model, compare the predicted labels to the true labels provided in the MNST database. Look at the error rates as the number of pixels used increase.
9. From the results of running the model on the training data, identify the optimal pixel count and solver
10. Test out the model on the test data
11. Repeat with one digit at a time. This requires the same steps, but instead use specific columns of X rather than averaging the entire matrix. Then the error calculation will be based on how well the model knows if the image is the specified value

4 Computational Results

4.1 Sparcity and relevant pixels

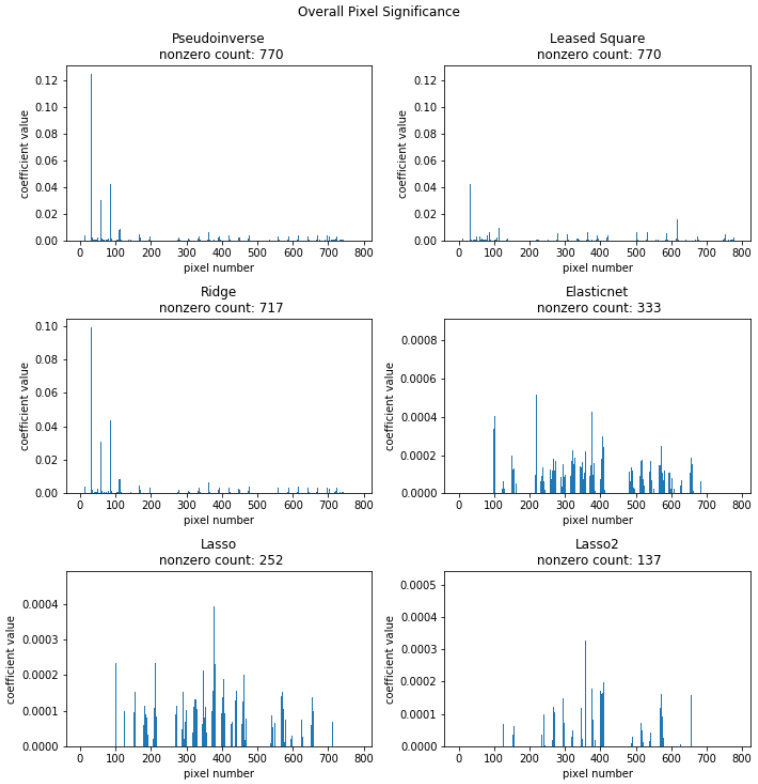
The colormap plots below highlight relatively high values in the coefficient matrices created from using different solvers. This initial view gives an idea of which pixels are most important for each model type and digit.



Notice that pseudoinverse, least square, and ridge look very similar. The straight lines indicate the relevant pixels are similar for all digits. Also there seems to be a strong distinction between relevant and non relevant pixels given the lack of medium hues.

ElasticNet Lasso and Lasso2 also have similar patterns of behavior in terms of relevant pixels, but differ from the first three. This makes sense because these models introduce L1 regularization. However they do differ in contrast between more and less significant pixels due to the different weights associated with the regularization. As expected, Lasso2, which penalizes the L1 norm the most, looks the most sparse.

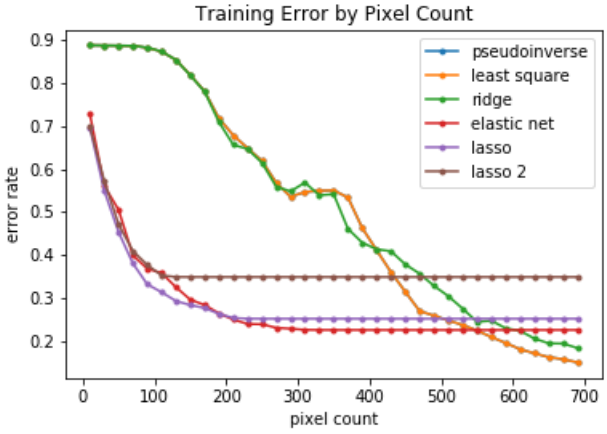
However, overall sparsity cannot be known exactly from the plot above to establish because the purple area may be zero or very small numbers. In the next image, average pixel significance (across all digits) is plotted for each model.



This image demonstrates the sparicty of each model. As expected we see that sparcity increases as the L1 norm is more highly weighted.

4.2 Error patterns by model

The below image shows the training dataset results in the model as the number of pixels used increases



From the plot above, the lasso model (which uses alpha=2) and elasticnet model perform well. The lasso2 model starts off well with the error dropping quickly, however it tapers off early on due to its higher sparcity. The pseudoinverse, least square, and ridge are far less sparse and cannot identify the digit well unless most of the pixels are used. However, these models end up with lower error overall when using all pixels. This is only marginal however. elasticnet and lasso reach 25% error by 200 pixels while it takes pseudoinverse, least square, and ridge around 550 pixels to reach that level.

This image demonstrates the trade off between accuracy and amount of data used. The most efficient models appear to be Lasso and ElasticNet.

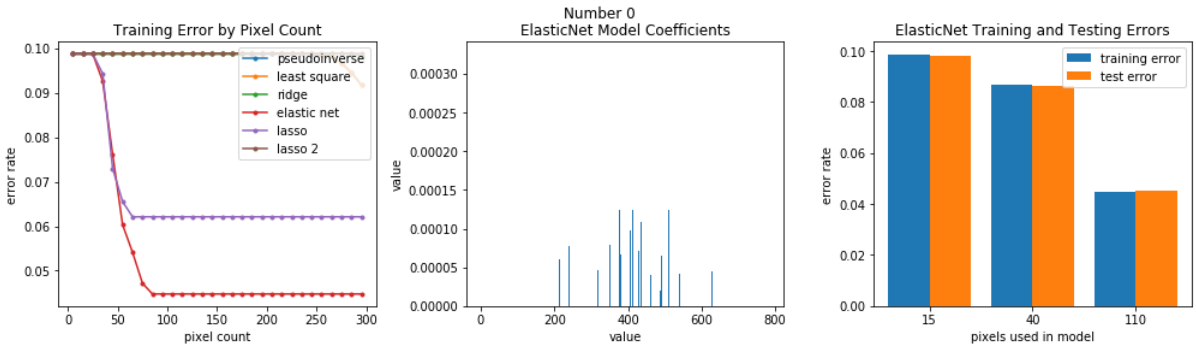
4.3 Test results

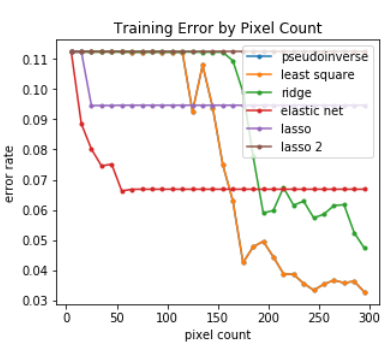
This plot below shows the errors of three of the models on the training and testing dataset

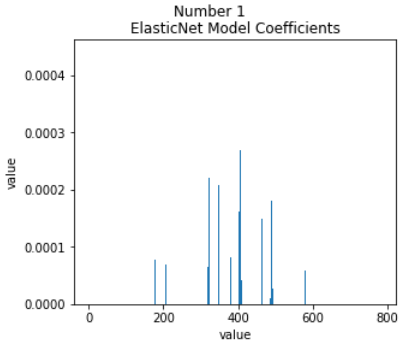
Here, Lasso performed the best and Elasticnet is close behind. Notice that for Lasso 2, which has a stronger penalty on the L1 norm error, the error stops dropping significantly after 100 pixels. This makes sense because as shown in the bar plot, Lasso 2 only contains 137 nonzero coefficients.

4.4 Specific number results

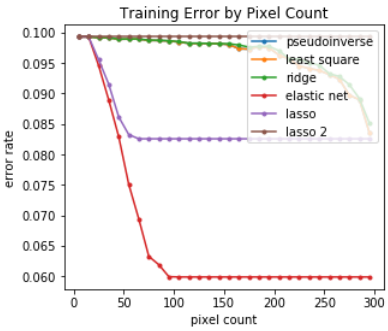
In the number specific model, the errors are lower overall, the highest being around 10%. The ElasticNet model seems to have the best error dropping behavior. Each digit has its own significant pixel patterns and error behavior. The numbers 0, 6, and 8 drop to 4-5% after around 50-100 pixels while the digits 4, 5, and 9 remain around 10%. The images below demonstrate the variability among the digits as well as the common patterns in error rates across models.

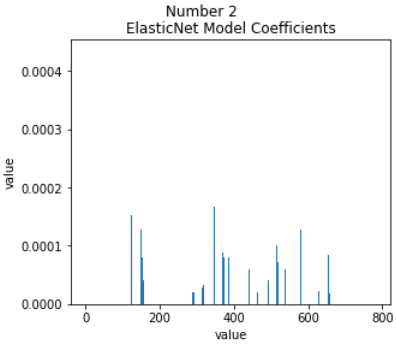


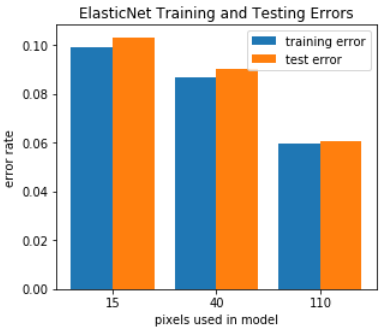

Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count

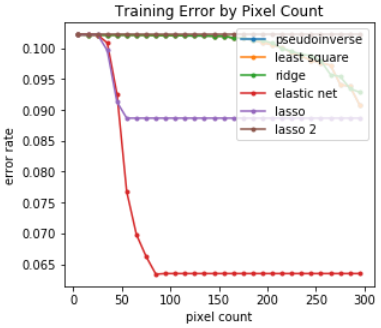

Number 1
ElasticNet Model Coefficients
value

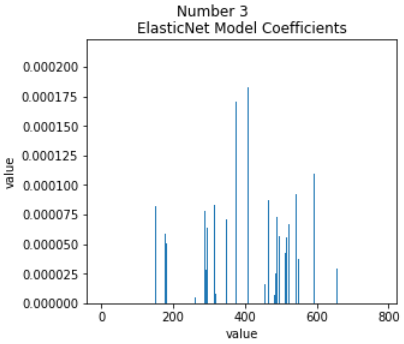

ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model

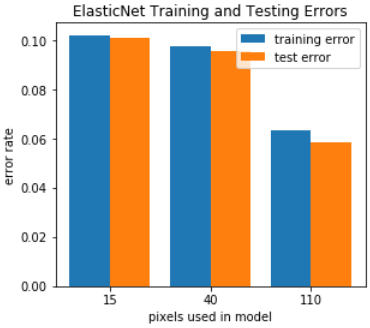

Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count

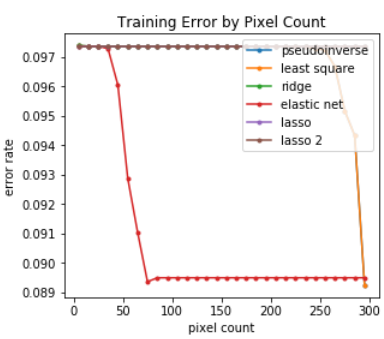

Number 2
ElasticNet Model Coefficients
value

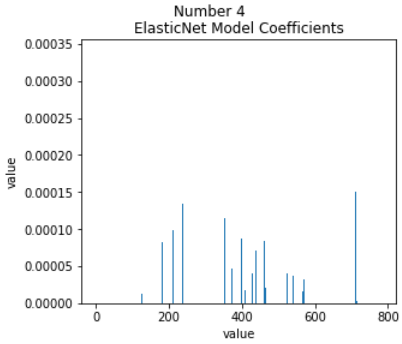

ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model

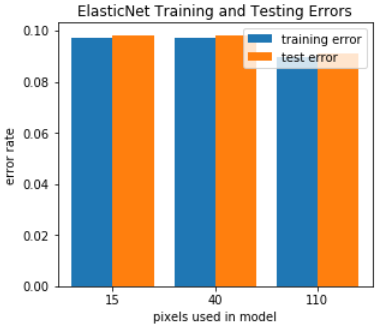

Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count

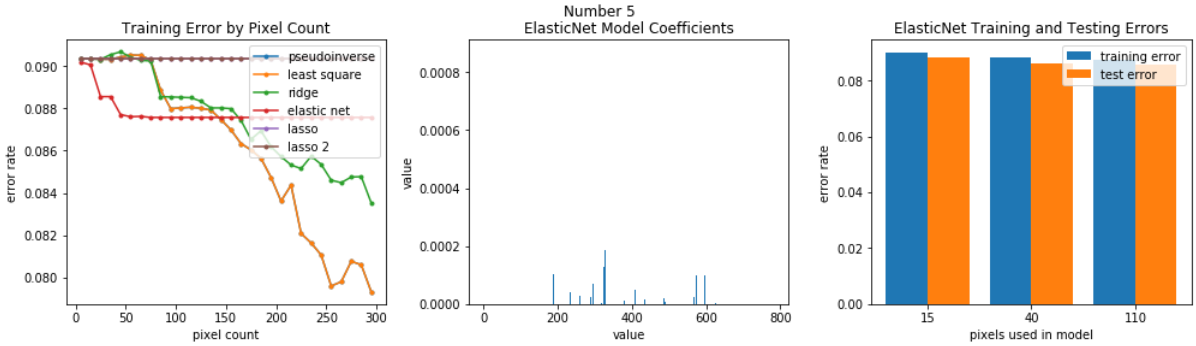

Number 3
ElasticNet Model Coefficients
value

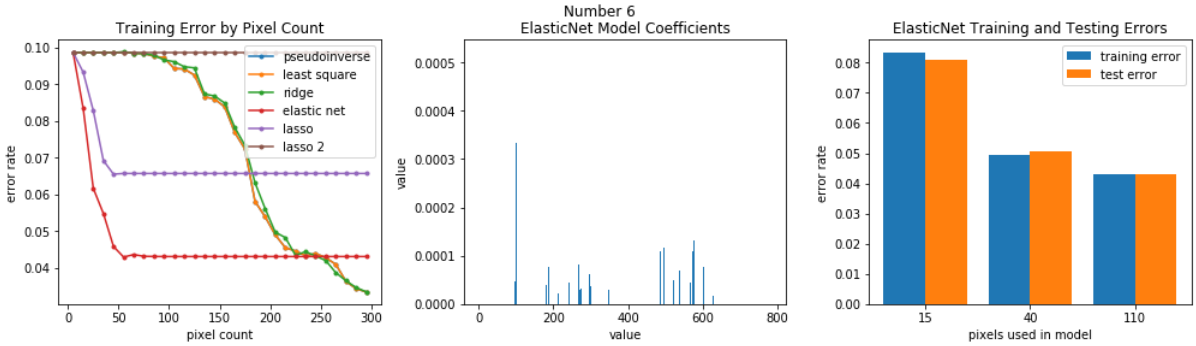

ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model

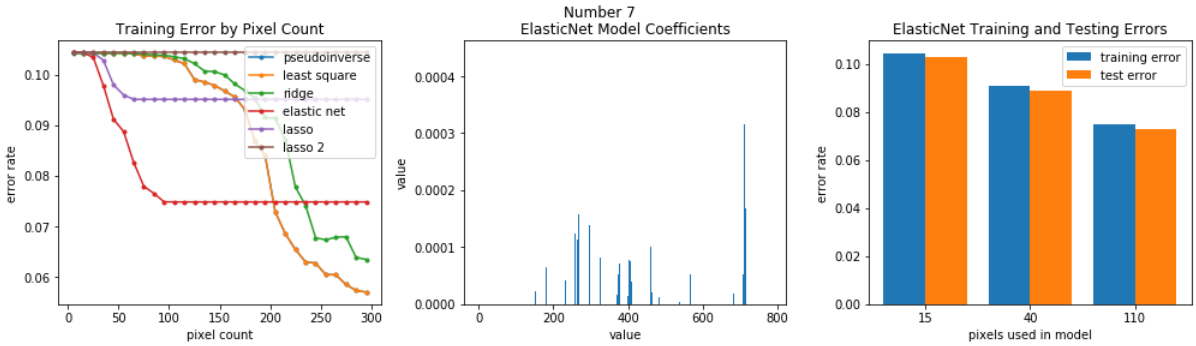

Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count

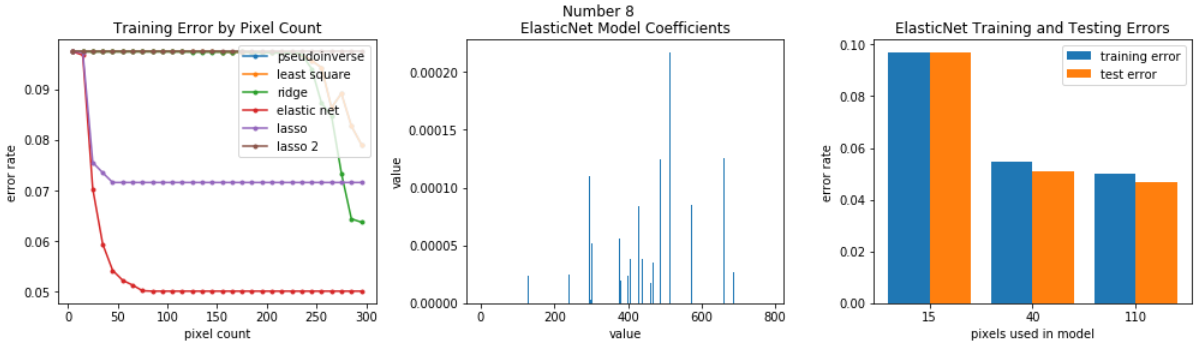

Number 4
ElasticNet Model Coefficients
value

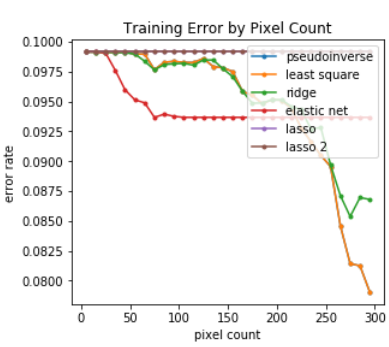

ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model

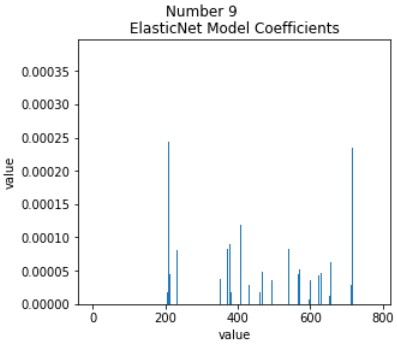

Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count
Number 5
ElasticNet Model Coefficients
value
value
ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model


Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count
Number 6
ElasticNet Model Coefficients
value
value
ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model


Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count
Number 7
ElasticNet Model Coefficients
value
value
ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model


Training Error by Pixel Count
pseudoinverse
least square
ridge
elastic net
lasso
lasso 2
error rate
pixel count
Number 8
ElasticNet Model Coefficients
value
value
ElasticNet Training and Testing Errors
training error
test error
error rate
pixels used in model





5 Summary and Conclusions

This experiment demonstrates several aspects of model selection. Most importantly, it shows how much models can vary in behavior, structure, and success simply due to the choice of regularizer. Model selection is dependent on the goal of the analysis. For example, if all pixels were used, a stronger weight on the L2 norm would probably have been preferred. However, in practical applications, data size and efficiency is extremely important, so achieving relatively similar success rate on a much smaller dataset is very valuable and often necessary.

6 Appendix A: PYTHON functions used and brief implementation explanation

1. np.linalg.lstrsqr - $AX = B$ least square solver
2. np.linalg.pinv - $AX = B$ solver using the pseudoinverse of A
3. linear_model.ElasticNet - $AX = B$ solver that uses L1 and L2 penalties with specified weights
4. linear_model.Ridge - $AX = B$ solver that uses L2 penalties with specified weights
5. linear_model.Lasso - $AX = B$ solver that uses L1 penalties with specified weights
6. gzip.open - opens the image and labe files from the MNST website
7. imshow - shows images of the handwritten digits

7 Appendix B MATLAB codes

See the following repository for the PYTHON codes used:

- https://github.com/rachel-carroll/computational-data-analysis/image_recognition_handwritten_digits