Regression and Model Selection

AMATH563: Homework 1

Dan Gorringe
dngrrng@uw.edu

April 21, 2020

Sec. I. Introduction and Overview

The prototypical MNIST dataset is used to demonstrate the introductory regression and model selection techniques. Lasso, ridge and Moore-Penrose pseudo-inverse are chosen as regression techniques. The findings come from a Python Notebook that can be found on my Github[1].

The main focus of this report is to explore promoting sparsity in regression techniques. The degree to which reduced pixel models can accurately model the whole system is investigated.

Sec. II. Theoretical Background

The MNIST dataset has 60k images to learn from, many more than the number of unique labels, 10. The system is overdetermined, so it is unlikely that there is a true solution therefore instead we search for a minimal error.

To promote sparsity, whilst minimising error, a penalty for a variety of norms on the loadings. The different models chosen assign different penalties.

Moore-Penrose is the simplest case, simply least square with no penalties. Lasso only applies a penalty to ℓ_1, where Ridge only applies a penalty to ℓ_2.

Solving $AX = B$ for the MNIST dataset is creating a linear mapping between vectorised images and their vector labels. This method does not exploit the information in adjacent pixels nor similarity of digits.

Sec III. Algorithm Implementation and Development

Regression Techniques

Lasso and Ridge were found to be linear models in the `sklearn` library. The ability to fit and predict with these is built-in. The pseudo-inverse approach is a little more hands-on: you simply create the inverse, a matrix. With the inverse matrix you use a post matrix multiply on your input matrix.

Cross Validation

It is of paramount importance to cross validate any model selected. Both Lasso and Ridge have built in cross validation. Using the k-fold technique an averaged pseudo-inverse is produced, in order to cross validate different iterations.

In order to get Lasso to work with a y-vectors, I needed to use MultiTaskLassoCV rather than LassoCV.

Label Prediction

Each model produces a 10 dimensional vector output, each dimension corresponds to the prediction of a certain digit. The highest of these values is taken as the overall digit prediction.

The dataset is not uniform, there are more of some digits than others. I have included a mean adjustment that should counter this.

Cleverer algorithms would account for the conditional probabilities between digits. There will be non-trivial optimisations where certain digits look more like others, and this can be exploited to more accurately label predictions.

Informative Pixels

Promoting sparsity decreases the number of non-zero values in the regression models. Finding the most important pixels is a slightly different task.

Each model explored creates a column of coefficients corresponding to every pixel. In increasing alpha values, promoting sparsity, the norms of the variables are minimised.

In order to find the most utilised pixels I look at the 2-norm of the pixel columns, and crucially weight them by the average weight of the corresponding pixel in the training set. Without the weighting I found that lots of the 'most important' pixels were very substantial when used, but infrequently caused consequence.

Sorting, and choosing the highest n, was how I choose the n-most important pixels.

Pixel Reduction

Seeing the ability of important pixels to make similar accuracy predictions with fewer variables I thought it would be interesting to construct new models using only these variables.

The previous models wasted computational complexity on pixels that we found to be unimportant. The scaling of models created using important pixels versus whole models with concatenated data was explored. Finding larger models prior to creating smaller seemingly optimised models is an interesting proposal and seems to be somewhat cheaty, but for loss in initial computation can be made up for if you have to regularly collect, maintain and store data.

[1] https://github.com/DanGorringe/AMATH573_inferringComplexSystems

Sec IV. Computational Results

Sec V. Summary and Conclusions

Question 1: Using Various AX=B solvers, determinate a mapping from the image space to the label space.

I choose to investigate the Lasso, Ridge and Moore-Penrose Pseudo-Inverse solvers. In Figure 1 the matrix coefficients from each solver can be seen. A qualitative measure of sparsity can be seen by the number of non average rows.

Accuracy of all three solvers is found to be around the same ballpark. 86.4%, 86.27%, and 85.42% for Lasso, Ridge and PInv respectively.

Question 2: By promoting sparsity, determine and rank which pixels in the MNIST set are most informative for correctly labelling the digits.

Promoting sparsity is the same as increasing the alpha value, the penalty for norms. For the Lasso model a variety of alpha values was explored.

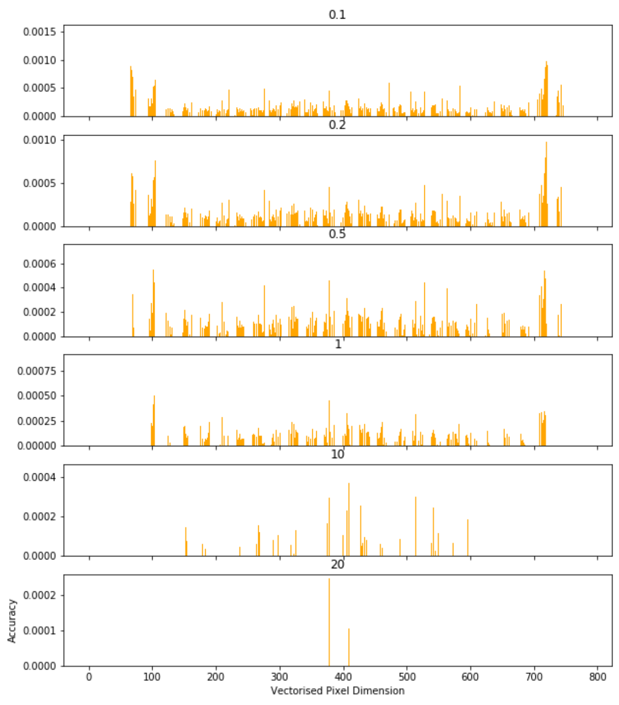

Figure 2: The norms of pixels for the coefficients in the Lasso model. With alpha values of 0.1, 0.2, 0.5, 1, 10, and 20.

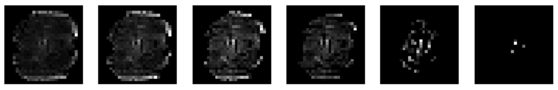

Figure 3: Image interpretation of model coefficients.

In Figure 2 we can see as the alpha value is increased the quantity of bars is decreased. as sparsity is promoted less and less pixels contribute to our model's predictions. Figure 5 plots these bars onto the pixels in the 28x28 MNIST image space.

Figure 4 shows the most important pixels for each regression model. It's interesting to see all models have some sort of preference for the very centre of the image. There's also a very interesting fascination with one of the very bottom rows.

Question 3: Apply your most important pixels to the test data set to see how accurate you are with as few pixels as possible.

The use of the learnt models using varying amounts of most important pixels can be seen in Figure

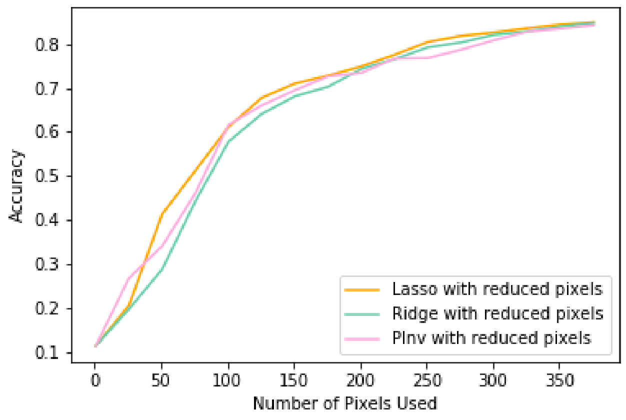

Figure 5: The accuracy of each model as we increase the number of most important pixels used in test batch data.

The current technique to find important pixels is greedy. It would be interesting to investigate non–greedy algorithms for finding the largest information gain.

I liked the idea of the information that the reduced pixels contain about each other. I explored this a little by relearning each model using the most important pixels. Although computationally intensive it was a fun experiment, and can be seen in Figure 6 and Figure 7 for all three regression techinques.

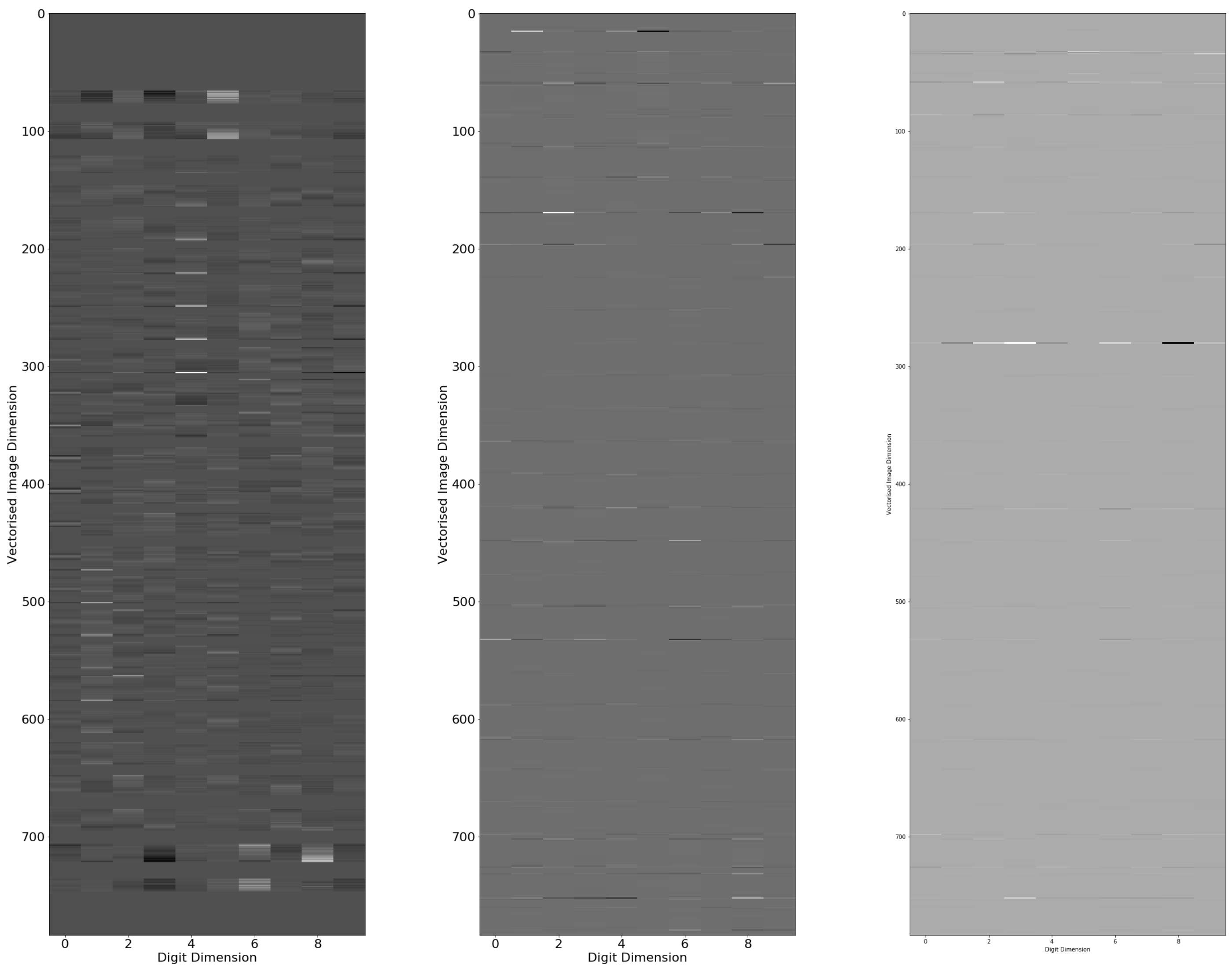

Figure 1: Different models were used to solve $\mathbf{AX} = \mathbf{B}$. From left to right: Lasso, Ridge and Moore-Penrose Pseudo-Inverse. For the first two their linear model coefficients are plotted, the pseudo-inverse matrix is shown.

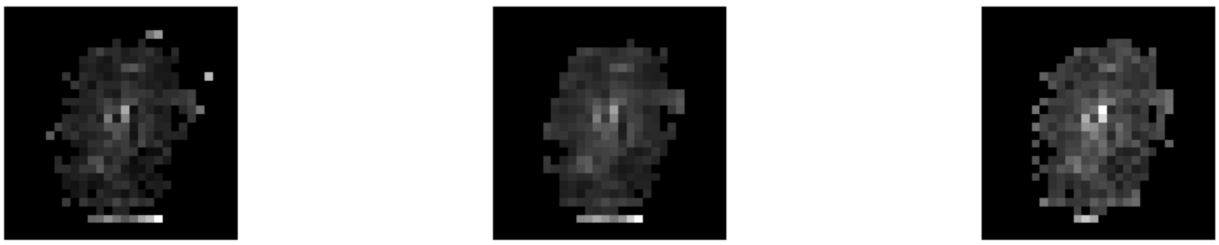

Figure 4: Visulisation of the 256 most important pixels for each of the regression models. From left to right: Lasso, Ridge and Moore-Penrose Pseudo-Inverse.

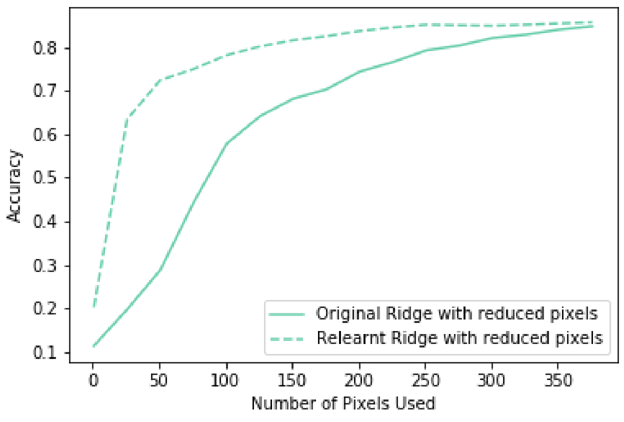

Figure 6: The accuracy of the Ridge model as we increase the number of most important pixels used in test batch data, against a model we train exclusively using these most important pixels.

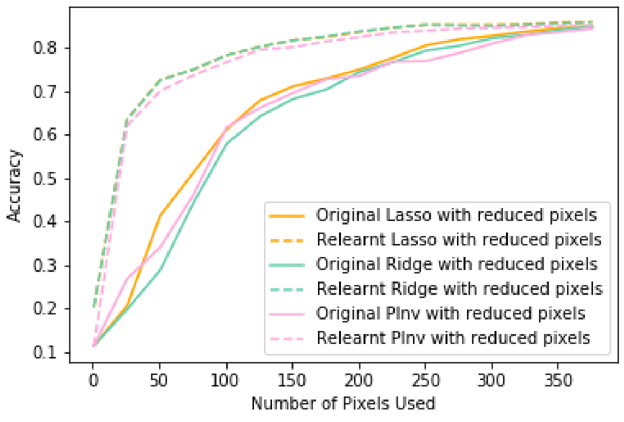

Figure 7: The accuracy of each model as we increase the number of most important pixels used in test batch data, against a model we train exclusively using these most important pixels.

Question 4: Redo the analysis with each digit individually to find the most important pixels for each digit.

Initially to find the most important pixels the coefficient norms were weighted by the mean of the whole dataset, a measure of probability of occurring. To find the most important pixels for each digit, means using the weights for each digit instead of the whole set.

Using and plotting the 64 most important pixels we find a lovely shadow of each digit, Figure 8. Here we can learn the significance of the the semi-mysterious bottom line found in Figure 4. Another quirk we find is the top right pertinent pixels for the digit 5 have made there way to the full model.

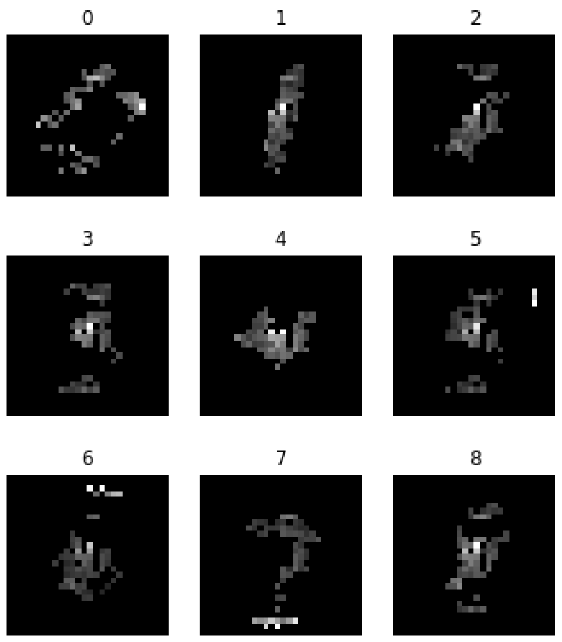

Figure 8: The 64 most important pixels for nine of ten of the digits classified using the Lasso model.

The accuracy on classifying the whole set is holistically pretty bad, as would be expected. For digit 5 however we see unique behaviour that for very limited pixels it can predict more accurately, Figure 9.

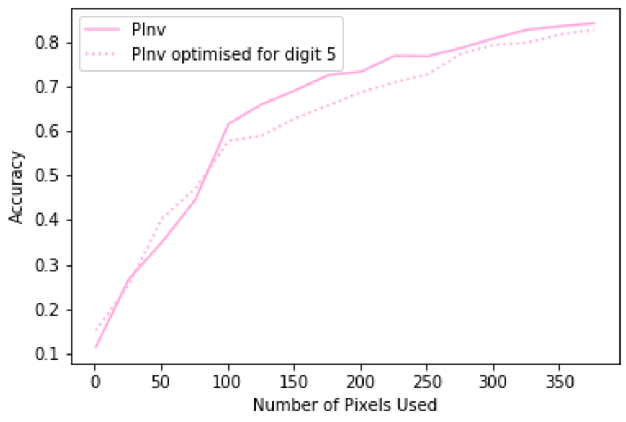

Figure 9: The accuracy of reduced pixel for PInv models learnt using all the digits versus the digit 5.