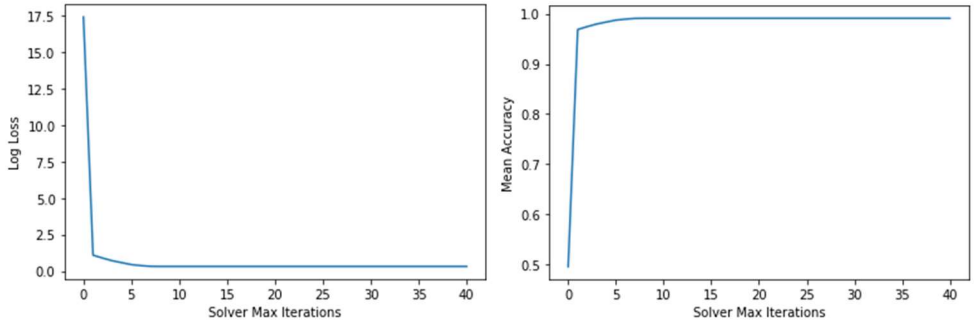
Project 1

Joseph Egan

October 2022

Part One

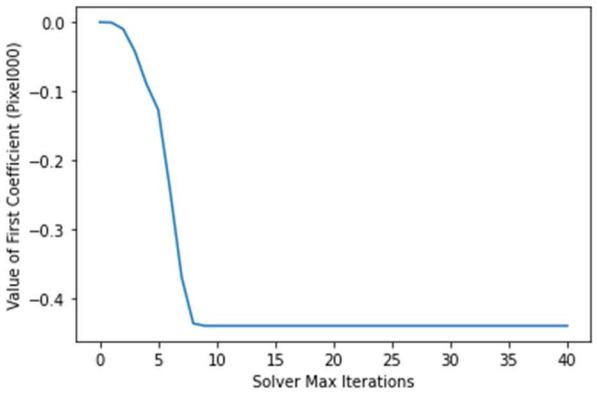
Question 1



Log loss for each model calculated at increasing max iterations (left) and accuracy for the same models (right)

The above plots show that the quality of the model generated when max_iter is limited drastically improves once the model is able to converge. When the max number of iterations is 1, the solver cannot make any updates to the weights and so accuracy is low and log loss is high. As the number of iterations allowed is increased, the solver is able to refine the model weights more until convergence at around 10 iterations.

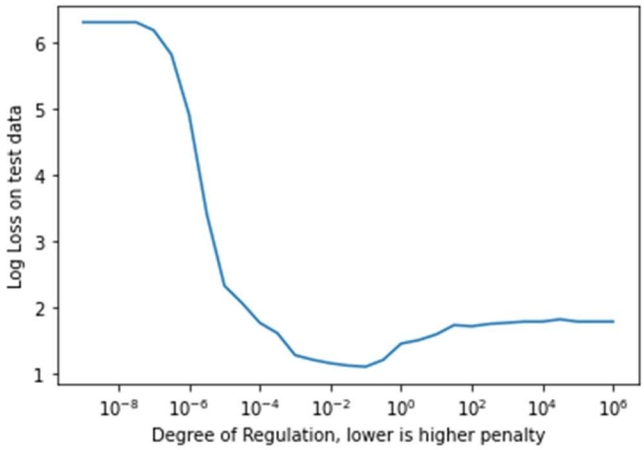
Question 2



Value of coefficient for first pixel (Pixel000) as iterations allowed increases

The above plot shows that as iterations allowed increases, the solver is able to approach an optimal value for the coefficient for this pixel after being initialized at 0. After the solver converges, the value of the coefficient is not changed.

Question 3



Log loss of ridge models with decreasing lambdas

When exploring regularization for this question, I evaluated both lasso and ridge regression techniques. It was found that L2 ridge regression was able to perform slightly better than lasso on the test data, and the optimum lambda was found to be 10.

I also evaluated 21 thresholds from 0 to 1 for the ridge model with lambda 10, and found that a threshold of 0.6 yielded a slightly higher accuracy on the test data than the default 0.5.

Predicted	0	1
True		
0	942	32
1	32	977

Predicted	0	1
True		
0	951	23
1	35	974

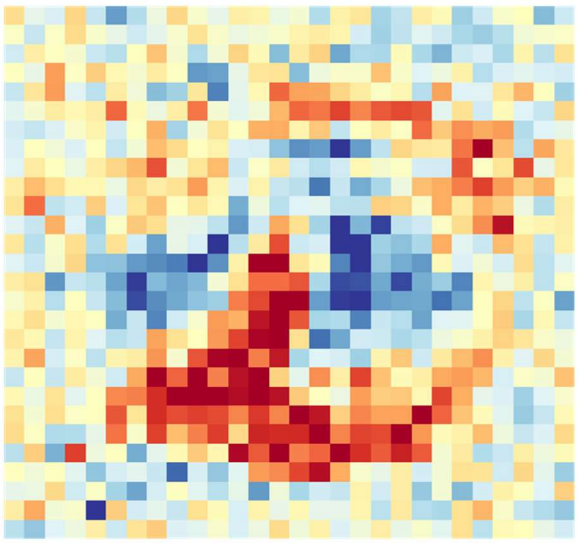
Confusion matrix for ridge regression models on test data with lambda 10 and threshold 0.5 (left) and 0.6 (right)

Question 4

False positives (left) and false negatives (right) from the final ridge model

For false positives, the images with incomplete loops, tight bottom loops, or that have a slightly tilted bottom loop are getting mislabeled by the classifier. For the false negatives, the bottom of the nine having a rounded loop causes the classifier to label them as 8s.

Question 5



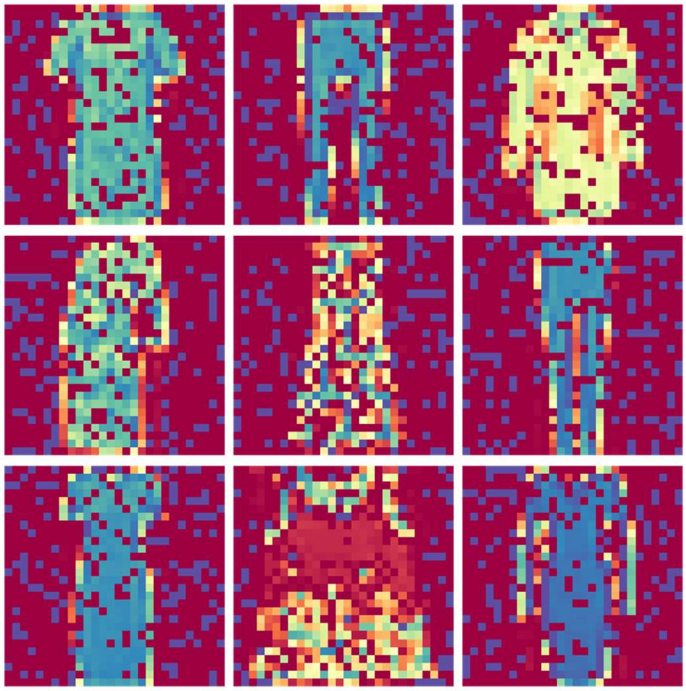
The values of each coefficient in the final ridge model displayed as a 28x28 matrix

In the above, blue pixels are closer to 1, yellow pixels are closer to 0, and red pixels are closer to -1. The easiest cluster to explain is the bottom left/center reds shaped like a reverse J. This is picking up 8s that are more likely to have values in this part of the image. To classify the 9s, the model then heavily weights values in the middle closer to the top of the number. This helps explain the false negatives in the above pictures – the model is classifying images with values in the red region as 8s, and without values as 9s.

Part Two

Exploration

Part two of this project was focused on a dataset containing trouser (1) and dress (0) images. I began by importing the data and looking at a sample of the images. This involved exploring different cmaps to find one that would best accentuate the differences in the images. I ended up going with 'Spectral'.



9 images from the data set. Images 2 and 6 are trousers and the rest are dresses

I then wanted to check how balanced the data set was – how many dresses vs trousers. This turned out to be exactly 50% for each.

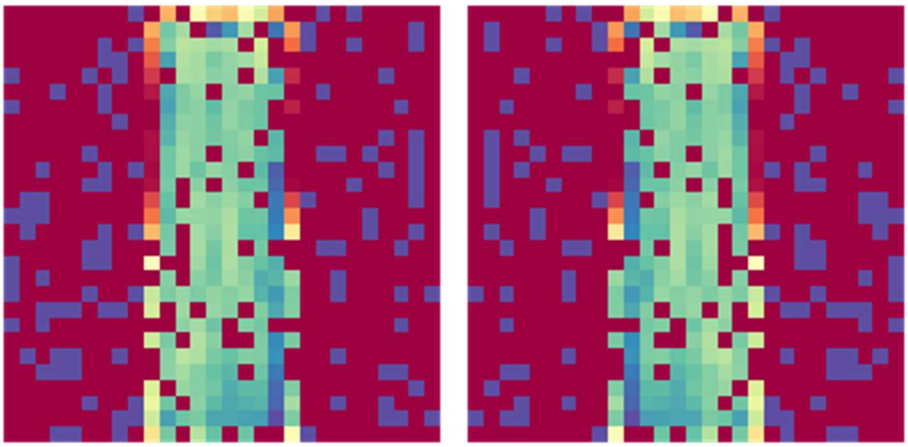
Building Models

The above being done, I moved on to training a simple logistic regression classifier to the data with 5-fold cross validation and no regularization, and obtained these results:

Model	Accuracy	Log Loss
Basic Logistic Regression	0.9263	2.5443

For this, and all other future models tested, I split the training data into 75% training, 25% validation/testing, and all quality metrics are based on results achieved from the validation set.

Before making any changes to the model, I wanted to check if doubling the data by flipping the images would add any power to the classifier.



Original (left) and flipped (right) images

I retrained the same model on the new, larger dataset and obtained these results:

Model	Accuracy	Log Loss
Basic Logistic Regression with double dataset	0.9391	2.1011

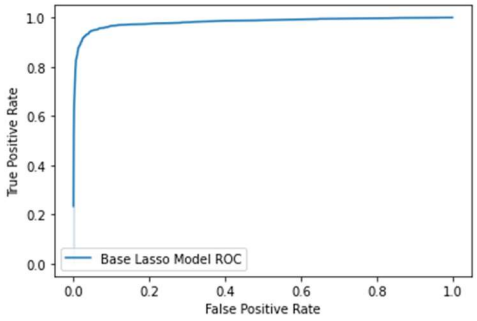
After seeing the slight improvement, I decided to operate with this doubled dataset going forward. The next step was testing if regularization would improve testing error. I did this by creating both lasso and ridge models and evaluating them at different lambdas. I achieved slightly better results overall, and even better results with lasso:

Model	**Accuracy**	**Log Loss**	**Optimal Lambda**
Lasso	0.9492	1.7557	10
Ridge	0.9490	1.7615	316

I decided to select the lasso model as the model to move forward with for two reasons:

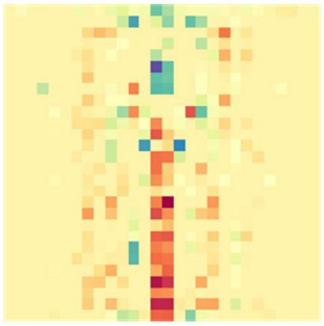
1. The results were slightly (though probably insignificantly) better than the ridge model's
2. It made the model simpler by driving coefficients to 0

For the lasso model, I then tested 21 thresholds from 0 to 1, and found that 0.5 was still the best threshold.

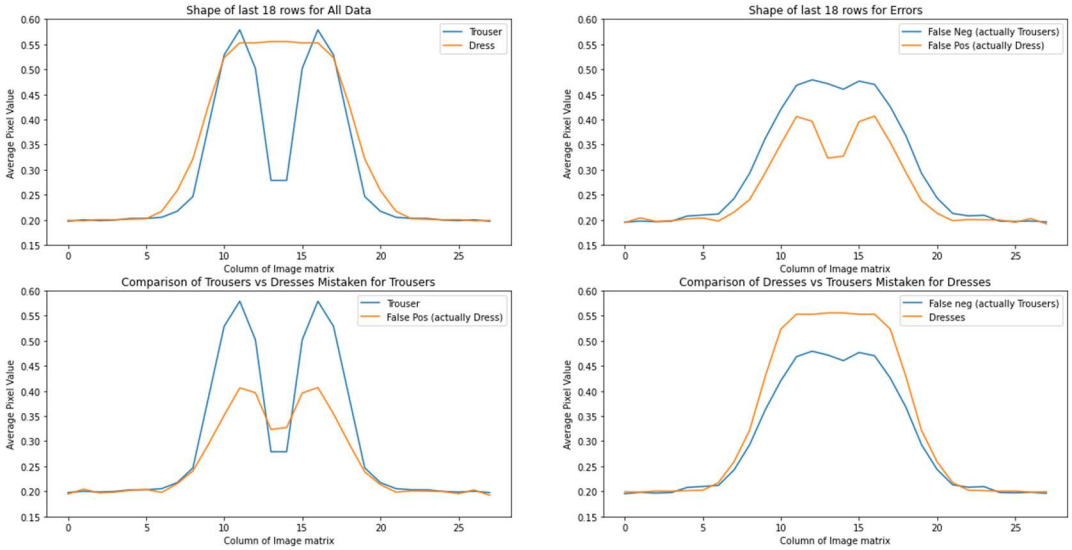


ROC curve for base lasso model

I then plotted the coefficients of the model so that I could verify my assumptions (it's based on the gap in the legs) about how the classifier was classifying the images:



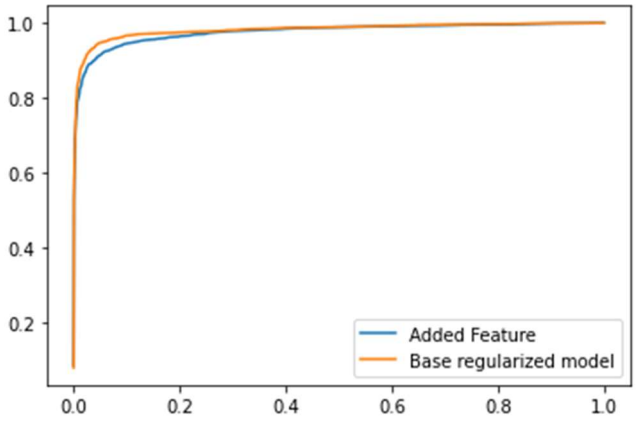
I also created several plots that compared the average values of each image in each column below (greater than) row 10. I was trying to get a sense of what each type of image looked like on average for all data, and false negatives/positives:



The above charts intuitively made sense – the false positives were dresses that looked like pants – they had lower values where the gap in trousers would be, and the false negatives were trousers that looked like dresses, they had little or no gap. Creating new features to capture this was difficult, but I focused on the observations that the false positives had lower peaks and less of a drop off in value in the gap region, and the false negatives overall had lower values than the dresses. For each image, I calculated average values for a left leg region (columns 9, 10, 11, 12), a gap region (13, 14) and a right leg region (15, 16, 17, 18). I then added two new features to the data. The first was 1 when the image had the characteristics of a false negative based on the relationships between the average values in each of the three regions, and 0 otherwise. The second was 1 when the image had the characteristics of a false positive, and 0 otherwise.

I then re-fit our best lasso model to this new dataset, and achieved an accuracy of only 0.9325, less than our non-new feature data. I was surprised by this since the model could have just turned off these

features if they were not helping. I assumed that the model was over-fit to the training data with these features, and with more time I could have tested across several lambdas again.



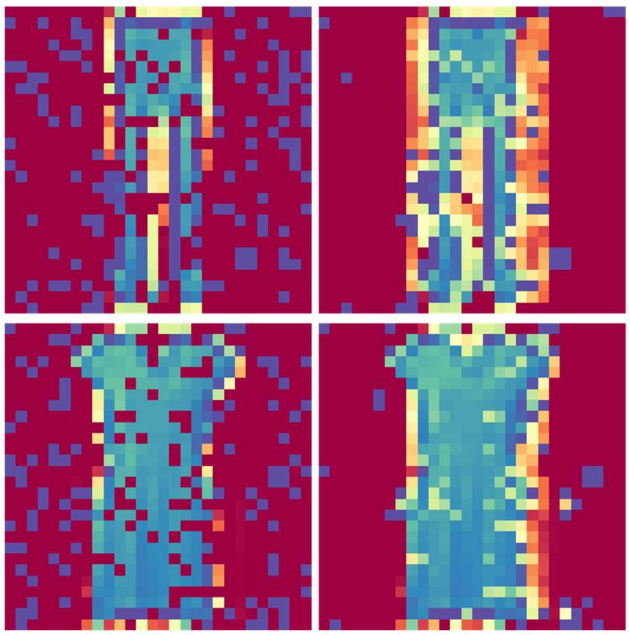
ROC curves for the base lasso model and the same model trained on additional features

Predicted	0	1
True		
0	2877	125
1	180	2818

Predicted	0	1
True		
0	2866	136
1	270	2728

Confusion matrices for base lasso model (left) and same model trained on additional features (right)

The final feature modification that I tried was creating a function that 'cleaned up' the images. I focused on filling in 'holes' in the images by averaging the surrounding pixels. I also tried to expand regions with values close to 0 in an attempt to accentuate the trouser gaps.



Before (left) and after (right) for two images sampled

After transforming the data in this away, I fit the base lasso model to the new data and achieved an accuracy of 0.9445, slightly worse than on the unmodified (but doubled) data.

Predicted	0	1
True		
0	2877	125
1	180	2818

Predicted	0	1
True		
0	2870	132
1	201	2797

Confusion matrices for base lasso model (left) and same model trained on 'cleaned' data (right)

Conclusion

Overall, the only modification that I made that had positive impact on the accuracy of the model was doubling the dataset. When testing the other modifications, I was able to achieve different results based on tweaking parameters within the functions. So, with more time and more scalable code, it's possible that I could have found the right parameters to make useful modifications/features. My best model ended up being a lasso regression model with lambda of 10 trained on 75% of the doubled training set and evaluated on the remaining 25%:

Model	**Accuracy**	**Log Loss**	**Optimal Lambda**
Lasso	0.9492	1.7557	10

When this model was retrained on the entire dataset, the predicted probabilities on the test set resulted in an accuracy of 0.9420.