COMPRESSED IMAGE RECOVERY USING LASSO REGRESSION

RYAN MOUW

Applied Mathematics Department, University of Washington, Seattle, WA
rmouw@uw.edu

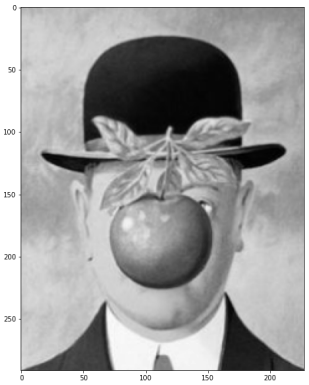
Abstract. The task is to recover an image given a limited number of pixel observations. In this problem we consider the painting, The *The Son of Man*, by Rene Margritte. The DCT coefficient analysis is performed to determine compressibiliity of the image. The images are then reconstructed after losing random pixels with Lasso regression in conjunction with convex optimization.

1. Introduction and Overview

In this problem we are given *The Son of Man* painting as seen in Figure 1a. The first goal is to compress the image by analyzing the image's Discrete Cosine Transform (DCT). The second goal is to reconstruct Figure 1c after removing 20%, 40%, and 80% of the pixels at random. To do this, Lasso regression is combined with the DCT of the image in an effort to take advantage of the sparsity in the DCT coefficients of natural images and the tendency for Lasso regression to favor sparse solutions. This is called sparse recovery.[4]

For computational purposes, the image is reduced to gray scale (Figure 1b) and also reduced in dimensionality (Figure 1c). The image that is used in this paper is $53x41$, resulting in 2173 total pixels. The lower resolution decreases the computational complexity of the problem. Ultimately, the sparse recovery is a success. The DCT analysis showed a lot of sparsity which indicates room for compression. The sparse recovery was also a success, with *The Son of Man* image (Figure 1c) being recognizable even after 80% of the pixel information had been removed.

(a) *The Son of Man*



(b) Gray scale

(c) Low resolution ($53x41$)

Figure 1. *The Son of Man* by René Margritte (Original, recolored, and rescaled)

Date: March 18, 2022.

2. Theoretical Background

The problem involves investigating the compressibility of *The Son of Man* painting using Discrete Cosine Transform (DCT). The DCT is a way of expressing a finite number of data points in terms of sums of cosine functions oscillating at various frequencies. Given a discrete signal, $\mathbf{f} \in \mathbb{R}^K$, we define its DCT as $DCT(\mathbf{f}) \in \mathbb{R}^K$ where;[2]

$$DCT(f)_k = \sqrt{\frac{1}{K}}\left[f_0 \cos\left(\frac{\pi k}{2K}\right) + \sqrt{2}\sum_{j=1}^{K-1} f_j \cos\left(\frac{\pi k(2j+1)}{2K}\right)\right] \tag{1}$$

The DCT is the equivalent of taking the real part of a Fast Fourier Transform. I.e., constructing the signal as a sum of weighted cosine functions. The inverse DCT (iDCT) then transforms the DCT back into the original signal, $\mathbf{f}$.[2]

The machine learning technique used in the image recovery is Lasso Regression. As with a multiple regression problem, we want to find a function that takes an input vector $\hat{x}$ approximates the observed output $\hat{y}$, i.e.;[8]

$$\hat{f}(\hat{x}) = \sum_{j=0}^{J-1} \hat{\beta}_j \Psi_j(\hat{x}) \approx y(\hat{x}) \tag{2}$$

Where $\Psi_j(\hat{x})$ are the features and $\hat{\beta}_j$ are the sought coefficients. Lasso regression is contrasted with simple linear regression in that a 1-norm (L1) regularization parameter is added to the loss function. In Lasso regression we want to find $\beta^\star$ such that;[8]

$$\beta^\star = argmin_{\beta \in \mathbb{R}^N} \quad \frac{1}{2}||A\beta - y||_2^2 + \lambda||\beta||_1 \tag{3}$$

Where A is the DCT of the image, β are the sought coefficients, and λ is the normalization parameter. The L1 regularization parameter addition results in solutions that are sparse, as compared to L2 (2-norm) regularization. The key connection between the DCT of natural images and Lasso regression is sparsity. DCT coefficients of natural images tend to be sparse and Lasso regression favors sparse solutions. This process is called sparse recovery and is a powerful denoising tool.[6]

3. Algorithm Implementation and Development

First the DCT and iDCT matrices are constructed. The DCT and iDCT matrices are created by applying Equation 1 to the entries of the identity matrix that is the same size as the image in question. This allows us to multiply the identity DCT and iDCT with any image that is same size as the original image. The result will be the same as if we took the DCT and iDCT of the original image. This modular approach will make the code simpler when we remove random pixels from the original image in later steps. The DCT matrix is then multiplied by Figure 1c after it has been flattened. The resulting vector holds the values of the DCT coefficients and can be seen in Figure 3.

The second problem is to recover the images after M random pixels are lost. M random pixels of Figure 1c are multiplied by zero and the image is then flattened. The resulting vector is then equivalent to the vector y from Equation 3.The same entries that were set to zero in y, are set to zero in the iDCT matrix that was created in the prior step. The resulting matrix is now equivalent to A in Equation 3.

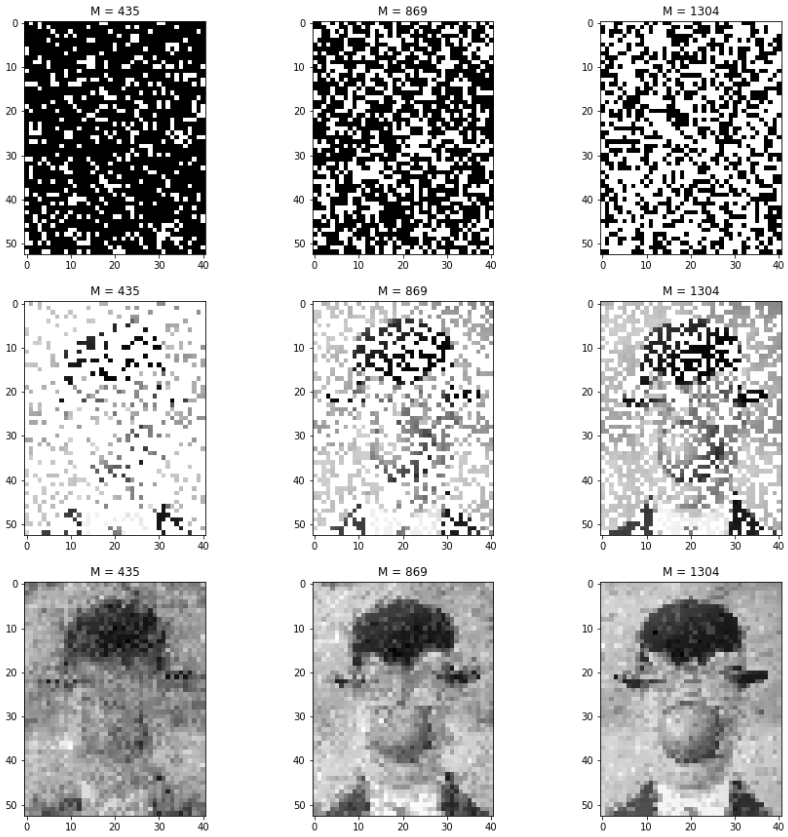


FIGURE 2. Image reconstruction after M pixels are lost

Instead of using a typical Lasso regression package to solve this optimization problem, CVXPY is used.[3][1] CVXPY is a convex optimization solver. This package allows the user to define an objective function to minimize along with any necessary constraints. As a result of using CVXPY, the hyperparameter λ, which is usually tuned using Cross Validation, is set equal to 1. In this case we want to solve the optimization problem;

$$\begin{aligned} &minimize_{x\in\mathbb{R}^N} \qquad ||x||_1 \\ &Subject \quad to \qquad Ax = y, \end{aligned} \tag{4}$$

The resulting x is then the DCT matrix that when multiplied by A, will return the approximation to y. To find the approximation of the original image, Figure 1c, instead of multiplying Ax to find y, x is multiplied by the iDCT of the identity matrix created in the first step. The result is an image that should resemble Figure 1c.

Figure 2 demonstrates the process. The top row of Figure 2 shows an example of which pixels will be removed at random. The middle row shows the image after the pixels have been removed. The bottom row shows an example of the reconstruction. This process is repeated two more times, each time selecting M pixels at random and the results are shown in Figure 5.

The matrix data manipulation was done using Numpy.[5] The plotting in Figure 3 used Matplotlib.[7] The DCT of Figure 1c was found using the fftpack from Scipy. [9]

4. Computational Results

The absolute value of the DCT coefficients for Figure 1c are plotted in Figure 3. As seen in Figure 3, there are few large coefficients, and many nearly zero coefficients. This is an indication that there is a lot of room for image compression. This is also an indication that Lasso regression will be a powerful tool in compressing the image.

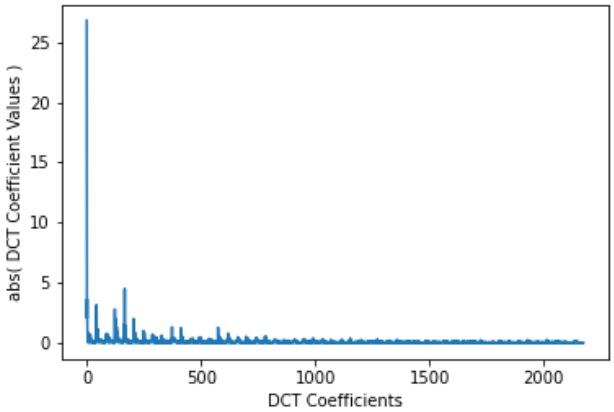


Figure 3. Absolute Value of DCT Coefficients

The images are then reconstructed using different proportions of the DCT coefficients. Figure 4 shows the reconstructed images after thresholding the DCT coefficient values so to keep 5%, 10%, 20%, and 40% of the largest DCT coefficients. As seen in Figure 3, with only 5% of the largest coefficients, the *The Son of Man* painting is recognizable. With 40% of the coefficients, the image is very similar to that of the original low resolution image (Figure 1c).

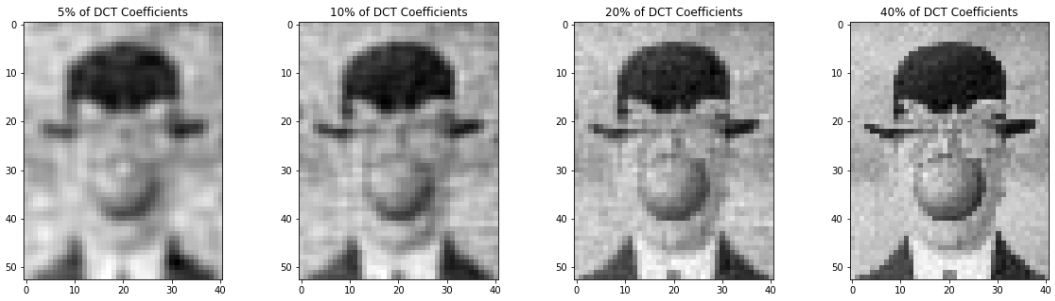


Figure 4. Figure 1c reconstruction using different proportions of the largest DCT values

Figure 5 shows the solutions to the convex optimization problem described in Section 2 and Section 3. The rows of Figure 5 correspond to different values of M and the columns represent the three separate trials. As Figure 5 shows, the reconstructions resemble Figure 1c quite well with just a fraction of the original pixel information. The reconstructed images in Figure 5 are noisier, especially in areas that were relatively smooth, like the periphery. The detail of the apple is also quite lost, even when $M = 1304$, but in general, the image is recognizable.

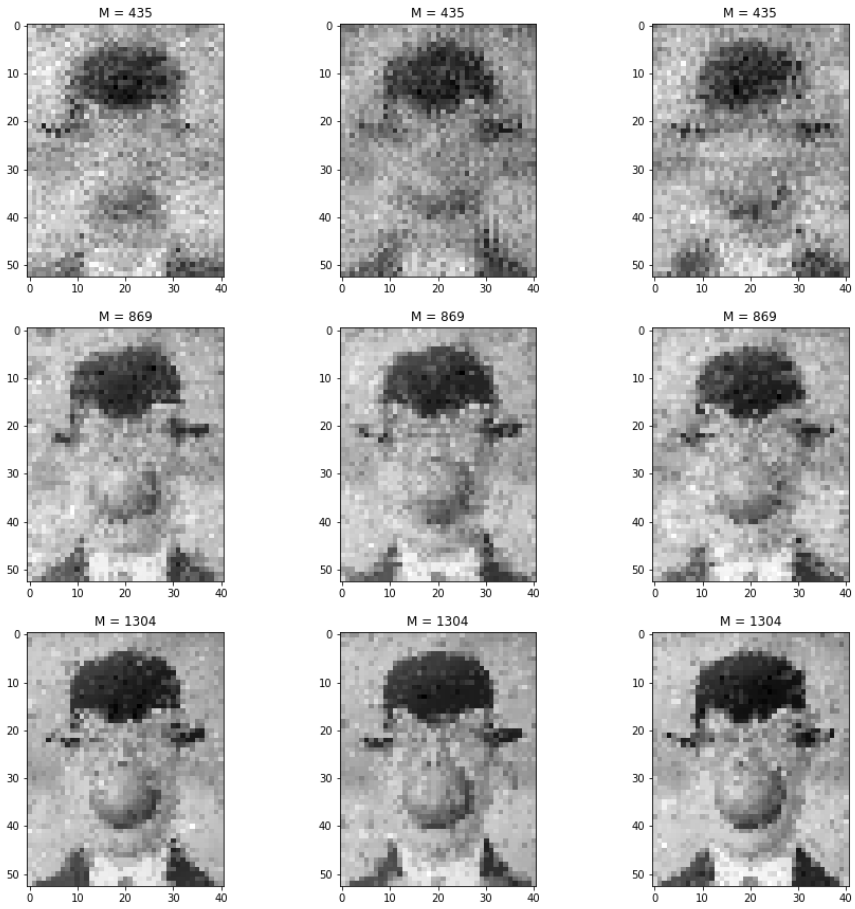


FIGURE 5. Three trials of Figure 1c reconstruction after M random pixels are lost

5. Summary and Conclusions

A fraction of the DCT coefficients were needed to make the image recognizable. This seems to be common in natural images as they tend to have less random noise. The largest difference between Figure 1c and the 40% DCT coefficient reconstruction in Figure 4 is that the reconstruction carries more noise. The original image is smoother in areas of homogeneous coloring, like the hat or the periphery.

Overall the combination of DCT and Lasso was able to reconstruct the images quite well. The image did lose detail in certain areas, like that apple, but the overall image is quite recognizable, showing that the combination of DCT on natural images and Lasso regression is powerful.

Perhaps further exploration could combine Lasso regression and Ridge regression to both remove the random noise from pixel loss, while maintaining as much detail as possible in the image, such as the area around the apple.

As a fun addition, we reconstruct another "Unknown Image." This image can be seen below.

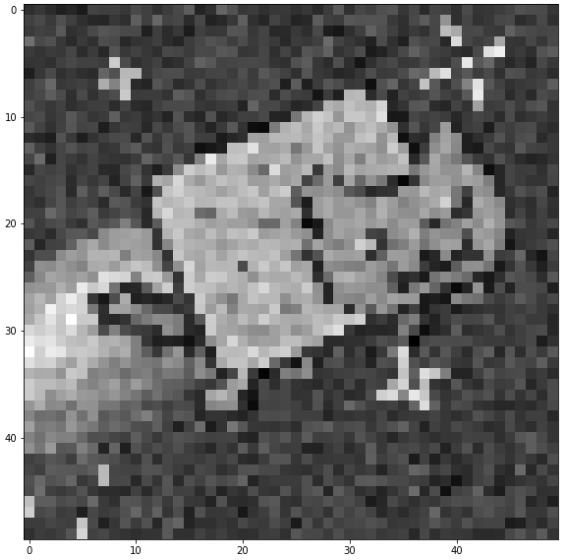


FIGURE 6. nyan cat

6. Acknowledgements

Prof. Hossieni's lectures were heavily relied upon for the DCT theory. [6]

References

[1] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[2] N. Ahmed, T. Natarajan, and K. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[3] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[4] A. C. Gilbert, Y. Li, E. Porat, and M. J. Strauss. Approximate sparse recovery: Optimizing time and measurements. *CoRR*, abs/0912.0229, 2009.

[5] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[6] B. Hosseini. Introduction to sparse recovery, Mar 2022.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[8] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.