AMATH 582: HOMEWORK 5

VINSENSIUS

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

Abstract. A scaled down of the son of man image is used for image compression and image recovery tasks. The compression task is done via direct cosine transform (DCT) by keeping the top 5%,10%,20% and 40% of the coefficients. The image recovery task is done by sampling 20%, 40%, and 60% of the total points in the image randomly with independent and identically distributed probability. The result from the image recovery task is that depending on the locations of the sampling, it has different results of the recovered image. However, with the increasing number of sampling, the recovered image has better image resolution. Using the same image recovery algorithm, the algorithm is used to recovered an unknown image with measurements given.

1. Introduction and Overview

The idea of sparsity is introduced because in the real scientific experiments, the data-sets are often very sparse and it is compressible. Thus, it gives rise to field of compressed sensing, where the field explores the idea having small number of measurements but the sensor is located at the critical position such that the information extracted can be an approximation of the real measurement data [3].

By using the idea of sparsity, companies can save a lot of resources because they can do smaller set of measurements at critical locations and have the same accuracy as they would if they had done large set of measurements at random locations.

In this paper, we will apply the idea of sparsity through image compression and image recovery tasks. The image is a painting of work called *The Son of Man* by surrealist painter René Magritte in 1964 [1]. We will the scaled down of the image for the two main tasks to keep the computational cost down. For the image compression, the compression is applied while keeping top 5%, 10%, 20% and 40% of the coefficients of the discrete cosine transform (DCT) of the image. For the image recovery task, we will apply random M measurements, which is an r fraction of the N points of the image. We will use r = 0.2, 0.4 and 0.6. From the measurements, we will attempt to reconstruct the image via DCT and making use of convex optimization via package to solve the problem.

2. Theoretical Background

Direct cosine transform (DCT) is the same as Direct Fourier Transform (DFT) but only taking the real part of the DFT [9]. DCT is easier to deal with because it is real-valued transformation and usually used for lossly image compression, where the information are loss during the compression process. From the scipy documentation page, the formula of the 2D DCT used for this problem is [11] [7]:

$$y(k_1,k_2) = 4\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1,n_2) cos\Big(\frac{\pi k_1(2n_1+1)}{2N_1}\Big) cos\Big(\frac{\pi k_2(2n_2+1)}{2N_2}\Big) \tag{1}$$

Date: March 19, 2022.

While the inverse is given within the reference paper [7]. The $x(n)$ is the function or input in the time domain while $y(k)$ is the function or output in the frequency domain.

Since we are working with natural image, it has sparse DCT coefficients so the image is compressible with high retention of important information as we will see later in section 4.

Since we are dealing with sparse data sets, if we want to use least squares regression, we will need apply L1 regularization. This is because with sparse data, L2 regularization (Ridge) will make the small data becomes smaller and the big value data to much larger value. Thus, the regression will be skew if L2 regularization is applied instead of L1. Thus, linear regression with L1 regularization is better with sparse dataset and known as Lasso regression. Similar to Ridge model, Lasso model seek to minimize the function of form [10]:

$$\hat{\beta} = \text{argmin}_{\underline{\beta}}(||A\underline{\beta} - \underline{y}||_2^2 + \lambda||\underline{\beta}||_1) \tag{2}$$

where $\underline{\beta}$ is the coefficients and the term to be found, A is a matrix of basis functions (feature maps), which is cosines evaluated at each data points (pixels) and $\underline{y}$ vector of output.

For image recovery, since the image is a natural image, the matrix A will be very sparse and number of sampling (M) is less than the total number of points (N), the lasso minimizer problem reduces to purely L1 minimizer problem:

$$\hat{\beta} = ||\underline{\beta}||_1 \tag{3}$$

with $\lambda = 1$ with constraint of $A\underline{\beta} = \underline{y}$. In this problem, matrix A is defined to be $A = BD^{-1}$, where B is a matrix with M rows of N-by-N I, identity matrix that has been randomized. N is the number of points in a R-by-C image. D is a matrix of DCT coefficients that has the same size of image. $\underline{y}$ is the measurement vector defined as $\underline{y} = B\vec{F}$ where $\vec{F}$ is the vector representation of the image F. The minimizer problem with the constraint is an optimization problem that has theory outside the class and thus is solved using optimization package, CVX in python [4]. The coefficient $\underline{\beta}$ that is tried to solve is an approximation of the DCT of the image. So, to get back the vectorized form of the image, we need to apply inverse DCT matrix on the coefficients. The variable $\underline{\beta}$ is the same as the variable x used in Image recovery part of section 3.

3. Algorithm Implementation and Development

Python packages that are used:numpy [5], matplotlib [6], scikit-learn [8], cvxpy[4] and scipy [11].

In this paper, we will make use of the smaller scale of the image to do all the analysis within the paper. So, we will use the rescaled image as our working image. We will compute the working image by applying function from scikit-learn and the scale will be 0.18 of the original image, which is 53-by-41 pixels. Furthermore, we will be using grayscale colour scheme for ease of computation.

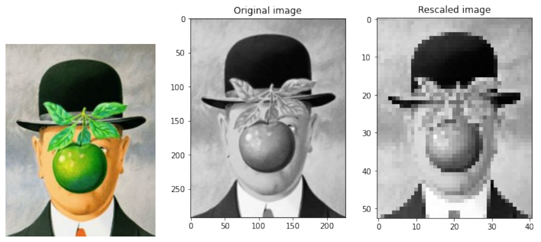
image Compression For image compression, the idea is that we will use top k% of the DCT coefficents of the the transformed image pixels in the frequency domain. We can generate the plot of the coefficients to see how the coefficients are behaving. Now, we can try to do compression by truncating the coefficients of the DCT(image). Firstly, compute the DCT of the image pixels.Then, sort the coefficient by their magnitude. Then, find the lower position of the top k% coeffient by rounding the index up such that that coefficient value be the threshold value and everything above that value is kept, while others are zeroed. After zeroing all the coefficient below the threshold, apply inverse DCT tranform to get the compressed image. Do it for top 5, 10, 20 and 40 percent of the DCT(image) coefficients.

Image Recovery For the image recovery, what we will do is to randomly picked M points where M is a fraction of N and $M << N$. The fraction r that is used are 0.2, 0.4 and 0.6. We will randomly selected M rows, based on the permutation of identity matrix I of $\mathbb{R}^{N\text{x}N}$, where N is the length of flattened size of the image. We will use the random permutation function from numpy package. We will call the permutation of M rows of I be matrix B which is in $\mathbb{R}^{M\text{x}N}$. Then we computed a vector y, which contains the measurement from the image, by multiplying matrix B with the working image. Compute A by multiplying B with the inverse of DCT matrix. Use the cvx package to solve the minimization problem of 1-norm of x, subject to problem $Ax = y$ describe in the section 2. Since x is the approximation of the transformation of image in the frequency domain, apply the inverse tranform of DCT matrix on x to get the image back. Do for each fraction for three times to see the difference for each random permutation and how the randomly selected rows will affect the result of the image recovery.

Unknown Image For the unknown image, use load function to extract vector y and B vector from the npz file for 50-by-50 pixel image. Then use the function that is created in the image recovery portion given vector y and matrix B to reconstruct the unknown image.

4. Computational Results

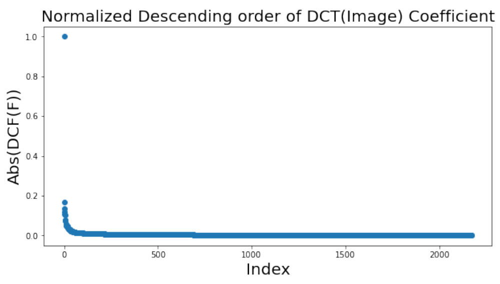
The section will discuss the results of the computation and relate it to section 2. The figure 1a shows the original colored image of Son of Image. The figure 1b on the left is the greyscale version of the coloured image (292x228), while on the right is the scaled down version of the original image (53x41). To make it clear, any word that is related to image or original image is referring to the rescaled image of the greyscale version of the original image. The only exception is during the discussion of image 6. In this case, the original image is referred to Son of Man image since the original image is based on painting. The discussed image 6 is referred as unknown image.



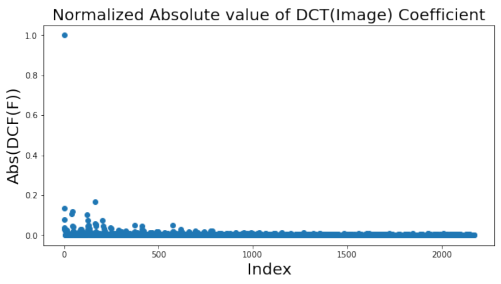
(A) (B)

Figure 1. Son of Man images of different colours and scales

As mentioned above, the scaled down version of the image is the working image and referred as image for the entire problem. The figure 2 shows the normalized DCT coefficient plots, one is in its original index, another is in descending order. From the descending order coefficients plot, it can be seen that the coefficients of DCT(image) are sparse with about the first 150 of the coefficients that are very important. This means that the image is compressible because the image can be represented by less coefficient and still be able to convey most of the information. From figure 3, we can see that doing 95% compression still retains most of the information from the original image such as the overall shape despite being blurry. As the compression decreases the image becomes clearer. Furthermore, with compression of 60%, the compressed image has most of the information about the subject of art.



(A)



(B)

Figure 2. Plots of normalized coefficient of the DCT(image)

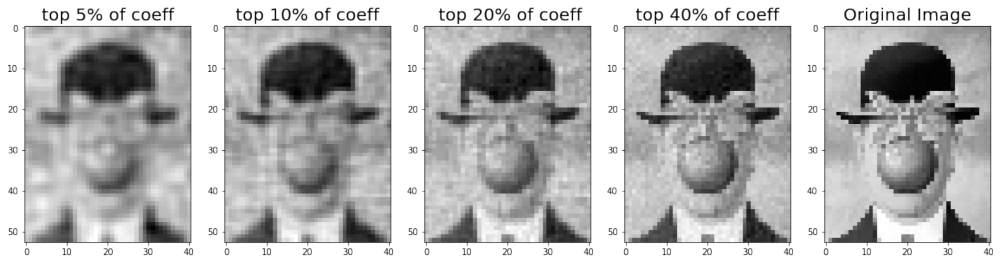
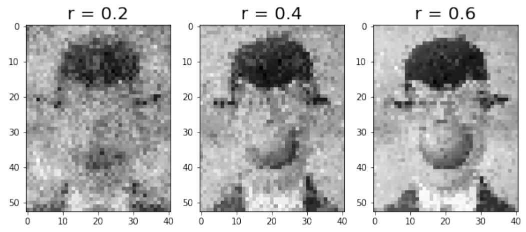
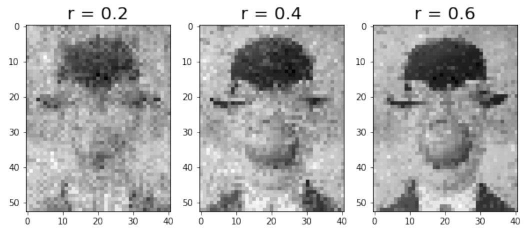


Figure 3. The image compression result by compressing 95%, 90%, 80% and 60% compared to the working (original rescaled) image respectively

Figures 4 and 5 show the image recovery based on different number of sampling based on the randomized rows of identity matrix with fractional number (r) of 0.2 , 0.4 and 0.6 over three trials. Since it is randomly selected from rows of identity matrix, the sampling is identically distributed. If we look at the particular r across three trials, we can see that depending on the location of the sampling, the image recovered will have different results. This is because if the sampling have a certain structure, it does not have enough wide range of frequency to excite the cosine modes. Thus, the image recovered will be very blurry. However, if the sampling has some incoherent structure, it will excite more cosines modes and thus giving better recovered image [3]. If we look at closely, r = 0.2 or 0.4 can be think of taking the top 20% or 40% of coefficients. But if we do random sampling instead of taking the top coefficients, it will be very hard to get the same results only by doing random sampling.



(A) trial 1



(B) trial 2

Figure 4. The sparse image recovery for sampling of different fraction of the total pixels of the image for first two trials. In each trial, r = 0.2 is on the left side, r = 0.4 is in the middle and r = 0.6 is on the right side.

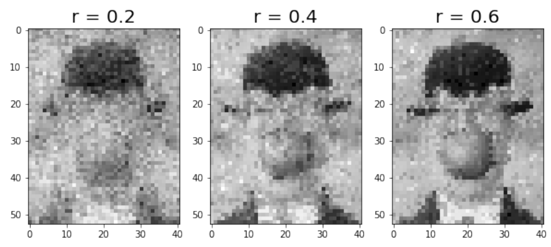


Figure 5. The sparse image recovery for sampling of different fraction of the total pixels of the image for the third trial. In each trial, r = 0.2 is on the left side, r = 0.4 is in the middle and r = 0.6 is on the right side.

Figure 6 shows the unknown image recovered based on some measurement of 50-by-50 pixel image. Based on the unknown image recovered, it is maybe compressed by about 60% based on what it is done on the rescaled son of man image.

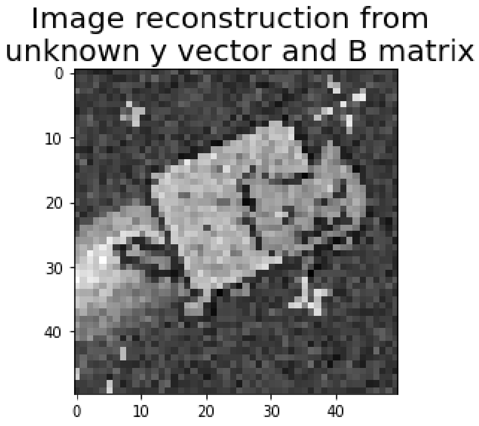


Figure 6. The image recovery based on the unknown vector y and matrix B.

5. Summary and Conclusions

In this paper, we have done tasks that are related to the idea of sparsity. Firstly, we did image compression and learnt that if it is applied on natural image in greyscale color , it has sparse DCT coefficients. As a result, the image is compressible and still retains most of the information about the uncompressed image. We found that compression somewhere around 80% is still fine before some of facial lines start to blur out. Then, we applied random measurements on the image with number of samplings be fractions of total pixels and tried to recover the original image. The image recovery is done by using L1 minimization of $||x||$ from linear problem $Ax = y$, which is an example of compressed sensing. Using the same image recovery algorithm, an unknown image is recovered based on some unknown measurement. The recovered unknown image is approximated to have about 60% based on the compression done on son of man image.

To further study the sparsity topic with this data set, we can study more about how to do effective sparse sampling such that we can recovered the original image with most sparse sampling. Another proposed idea is to use different basis function for image compression such as wavelets. To add on, we can define the accuracy of the recovered image against the original image or define the accuracy of the algorithm.

Acknowledgements

The author is thankful to Prof. Hosseni in teaching the material, providing the demo of the lectures and the helper code to get started on the problem. I would like thank Raymond Mead for pointing out the mistake in my understanding of the sampling of the image data for image recovery portion. I would like to thank Steve Brunton for his youtube video on example of FFT image compression using Python that helped me on my compression algorithm [2]. I would like to thank 482/582 discord community for providing helpful ideas for report writing. The discord community also helps me figure out other programming issue and guidance on the problem.

References

[1] Artincontext. "the son of man" magritte - an analysis of the famous apple painting, Feb 2022.

[2] S. Brunton. Image compression and the fft (examples in python), 2020.

[3] S. Brunton and N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

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

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

[7] J. Makhoul. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):27–34, 1980.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Vinsensius. Amath 582: Homework 1, Jan 2022. Unpublished.

[10] Vinsensius. Amath 582: Homework 2, Feb 2022. Unpublished.

[11] 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.