ECE 592 – Topics in Data Science
Project 4

Dror Baron; Fall 2019

Due: November 13, 2019

Administrative instructions:

1. For any clarification or doubts, the TA Hangjin Liu (hliu25 AT ncsu DOT edu) is in charge of homeworks and projects. She should be your first point of contact on homework- and project-related issues.
2. The project can be submitted individually, in pairs, or triples.
3. You should submit electronically through Moodle by midnight the day that the homework is due.
4. Your report should describe any mathematical derivations, responses to questions, results including any plots, and your MATLAB/Python code. Please justify your answers carefully.

Image compression via clustering: This project demonstrates how clustering can be used to compress images. An image compression system is comprised of:

- An input, $x \in \mathbb{R}^{M \times N}$. This is a real valued image of size M×N.
- An *encoder* maps x to a codeword f(x), which is comprised of bits. That is, f$(x) \in \{0,1\}^*$, where the asterisk denotes any finite length string over the alphabet (in this case the binary alphabet {0,1}).
- A *decoder* maps the bit string f(x) back to some real valued image, $\hat{x} = g(f(x))$.

Because we model images as real valued, it is unrealistic to be able to decode the original input image x perfectly from the encoded bits f(x), because there are infinitely many possible inputs. Therefore, we focus on *lossy compression*, where there is some loss or distortion between the input x and output $\hat{x}$. In particular, we use squared error distortion defined as follows:

$$D(x, \hat{x}) = \frac{1}{MN} \sum_{m=1,n=1}^{M,N} (x_{mn} - \hat{x}_{mn})^2,$$

where the subscript mn refers to row m and column n within the image.

We will perform lossy compression by partitioning the image into patches, clustering all the patches, and then all patches in the image that were mapped to the same cluster are mapped to the same representation patch. To make these steps more formal, we list them:

- Partition the input $x \in \mathbb{R}^{M \times N}$ into identically sized patches. To keep things simple, we will assume that each patch lies in $\mathbb{R}^{P \times P}$, where the patch length along each dimension, P, divides M and N evenly. For example, if P=2, then one patch in the corner of the image would contain x_{11}, x_{12}, x_{21}, and x_{22}. The patches will be non-overlapping, and so there will be MN/P^2 patches in total. (Hint: if using MATLAB, the reshape command may be helpful.)
- Stack the MN/P^2 patches together, and run some clustering algorithm on them. The output of the clustering algorithm will be C clusters, and each one will contain a representation patch.
- The actual transcription into bits involves scanning all MN/P^2 patches in the image, and for each one outputting the index of the cluster that corresponds to it. If we select C, the number of clusters, to be 2^k, then we can encode each of the indices using k bits.

This lossy compression method is called *vector quantization* (VQ) in the literature. The term vector refers to multiple pixels (P^2 could be 4 or 16, for example) in a patch all being encoded together, and quantization is discretization into a finite set of possible representation patches.

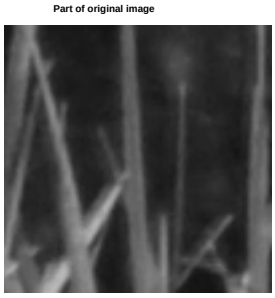
Information theory characterizes the best-possible performance limits of lossy compression. The study of these performance limits is called *rate distortion (RD) theory*. RD theory shows that there is a trade-off between the rate R, which is the average number of bits needed to encode each pixel,

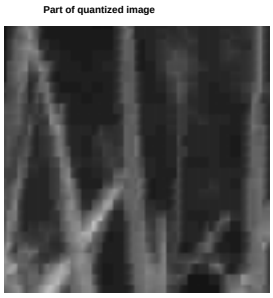
$$R(x) = \frac{1}{MN}|f(x)|,$$

where |f(x)| denotes the length or number of bits in the encoded string f(x), and the distortion D(.,.) is defined above. As the rate R is increased, the distortion goes down. Moreover, the relationship between R and D is convex.

1. **Load an image.** Select some relatively large image that you like, and load it. (Hint: if using MATLAB, you may want to use the imread command.)
   The size of the image may be inconvenient to process in terms of P dividing M and N evenly. To overcome this problem, please select min{M, N} to be a power of 2, and max{M,N} is a multiple of that power of 2. For example, M could be 512 and N could be 512*3=1,536. (I used M=N=2,048.) After choosing M and N, select a subset of the image of size M×N that "looks nice."
   Please provide a plot of your image, and code that loaded it, chose appropriate values of M and N, selected a subset of the image, and plots it. For example, you can see my image.
   Note: your image should be in gray scale, and not a color image with 3 color planes. In my example, I selected one of the three color planes and processed that part.

2. **Clustering and vector quantization.** Choose an initial patch size, P=2, resulting in P^2=4 pixels per patch. (This value of P will work well for the values of M and N chosen above.) Partition the image into MN/P^2 patches, and cluster the patches into C=16 clusters. The value of C was chosen by setting the rate R=1, meaning that each patch is encoded in RP^2=4 bits, and $C=2^4$=16 clusters can be encoded using 4 bits each. (Hint: in MATLAB, you can use the kmeans command.)
   Next, apply VQ to the image. You do not need to encode patches into bits (this would be tedious and not insightful). Instead, replace each patch in the image by the representation patch from the appropriate cluster.
   Please provide a plot of a small part of your image and the quantized version. Make sure to attach the code that produced these plots. My results for part of the grass in the image are included.





3. **Rate vs. distortion.** In this part, we wish to sketch the RD performance. To do so, choose a range of coding rates R. For each R, repeat the clustering and VQ of Part 2. For each rate R, compute the distortion D (.,.) between your original input x and its quantized version.
   Please provide a plot showing the trade-off between R and D. For example, you could plot R on the horizontal axis and D on the vertical axis. Please make sure to attach the MATLAB / Python code used to compute this plot. Please discuss the results.
   Note: For your rates, you could choose $R \in \{0.2, 0.4, 0.6, 0.8, 1\}$. That said, the important part here is to choose rates that provide an interesting RD comparison.
4. **Patch size.** In this part, you will vary the patch size. Whereas earlier you used P=2, now you will use larger P and rerun the rate distortion comparison of Part 3. You will need to again use some power of 2 for P; I found that P=4 runs slower than P=2, but the runtime is still acceptable as long as R is small enough. You may want to compare multiple (P, R) pairs.
   Please provide a plot showing how the RD trade-off varies for some values of P. Make sure to include your code. Did the overall RD trade-off improve, degrade, other? Please discuss your results, and try to explain what you think happened.
5. **Better compression.** Up to now, we have used an encoding technique that uses $C=2^k$ clusters, and the cluster index is encoded with k bits. However, it is likely that some clusters appear more often than others in the data. (In an example that I tried with k=8, the least popular cluster was used 93 times, while the most popular one appeared 2,994 times – quite a difference!) Similar to Morse code, it is possible to compress better by using fewer bits for popular clusters.
   In information theory, it is well known that one can encode a message with probability p using $\log_2(1/p) = -\log_2(p)$ bits. To quantify the potential savings in coding rate, we count the number N_i of times that cluster i was used, and then compute the probability p_i of that cluster by normalizing (dividing) N_i by MN/P^2, the total number of patches, i.e., $p_i = N_i/(MN/P^2)$. The average coding length required per patch is the *entropy* of clusters, $H = -\sum_i p_i \log_2(p_i)$, and the rate normalizes by the number of pixels per patch, $R=H/P^2$. (In my example, R=0.96 bits per pixel, which reflects a 4% reduction in the coding rate.) Actual encoding into bits is called *entropy coding*, and will not be pursued as part of the project.
   Please see what coding rates could be obtained with entropy coding for some of the examples you tried in previous parts. Discuss your findings.

More: Below are some possible directions for you to pursue. *Please select one of these based on your interests for full credit*. If you select more than one such direction, a modest amount of extra credit will be provided. (You may also want to pursue some of these as your individual project.)

- Implement K means yourself. Make sure that your code is well documented. It would be nice to compare your output to MATLAB or Python's standard function(s), as well as contrast run times.
- Read about some other clustering method and run it instead of K means. Please explain in detail what clustering method you chose, why, and how it affected overall compression performance.
- Suppose that we compute our clusters on a training image (or even multiple training images), and run them on a test image, which does not belong to the training set. Do you expect the error to be roughly the same, improve, deteriorate? Please put together relevant numerical results, present them, and discuss your findings.

Finally, feel free to chat with the instructor about other extensions to this project that you may want to pursue instead.