Implementation of the k-means algorithm with application to image compression

Steven Luu

Abstract

We implement the k-means algorithm from first principles and use this for image compression. The goal is to obtain a compressed image of a 512x512 pixel image. To achieve this, we instead apply the k-means algorithm on a 128x128 version of the original image for the purposes of faster compute time. The k clusters obtain from this task is then used to compress the large image. We also compare the resultant image to one which was obtained using singular value decomposition.

1 Introduction

The aim of this task is to compress a given image. Colored images are composed of pixels which contain 3 values referring to the pixel's red, green and blue (RGB) values. To compress the image, we aim to find a sufficiently small number of k unique RGB values, which we can use to approximate the original image. Let N denote the number of unique RGB values of the original image. If we can find a suitable value of k such that $k < N$, then we will have compressed the image by a factor of k/N. This is useful for memory/storage applications.

To achieve this, we will implement the k-means algorithm, which is an example of an unsupervised learning method. In general this algorithm aims to categorize a given dataset into k distinct clusters. Each data point is assigned to the cluster which it is closest to. This task is typically achieved by using some distance metric; a common metric is the classical Euclidean distance. Once a cluster has been assigned to each example of the dataset, the values of each of the initial k clusters are then updated by the average/mean of those assigned to it. The algorithm repeats this process until the difference between the two latest cluster updates is no greater than a specified tolerance value.

2 Results

The code used for k-means algorithm is contained in the file named 'k_means_algo'. We wish to apply this algorithm to compress a 512x512 pixel image. Direct application of the k-means algorithm to an image of this size is not desirable as it may take a considerable amount of time until convergence. Rather, we instead apply the algorithm to a 128x128 pixel sized version of the 512x512 pixel image; this reduction may be obtained by some convolution operations.

For this problem, we set $k = 16$. While this was arbitrarily chosen, the optimal value of k may be found by minimizing the error between the images obtained from k-means and the original. For example, the least mean squared error may be chosen as the error metric in this case.

For initialization, we chose $k = 16$ unique RGB values of the 128x128 pixel image and run the algorithm until convergence. The results are described in the following images.

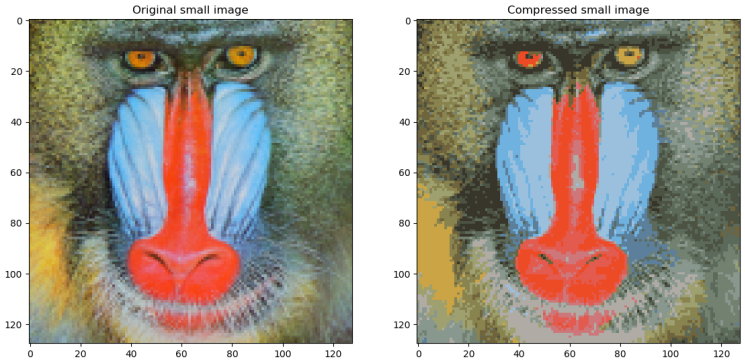

Figure 2.1: This figure illustrates the original small image on the left and the compressed small image on the right. The compressed image was produced using the k-means algorithm, with $k = 16$.

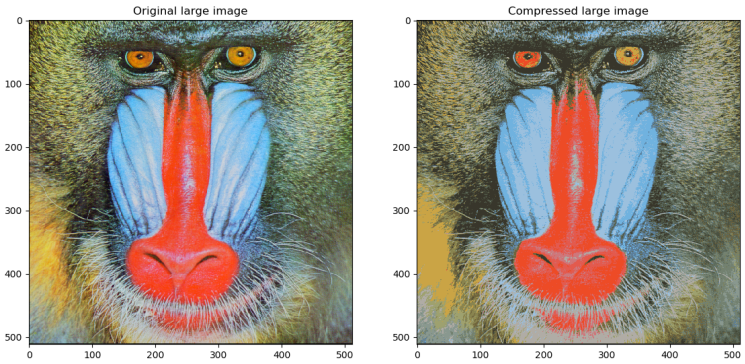

Figure 2.2: This figure illustrates the original large image on the left and the compressed large image on the right. The compressed image was produced using the k-means algorithm, with $k = 16$. From this figure we see that the compressed version of the large retains much of the main features of the original, while maintaining a 'crisp' image. Implementation of the k-means algorithm resulted in the image compression of the large image by a factor of 14401.

We compare the resultant image produced from the k-means algorithm to that obtained using

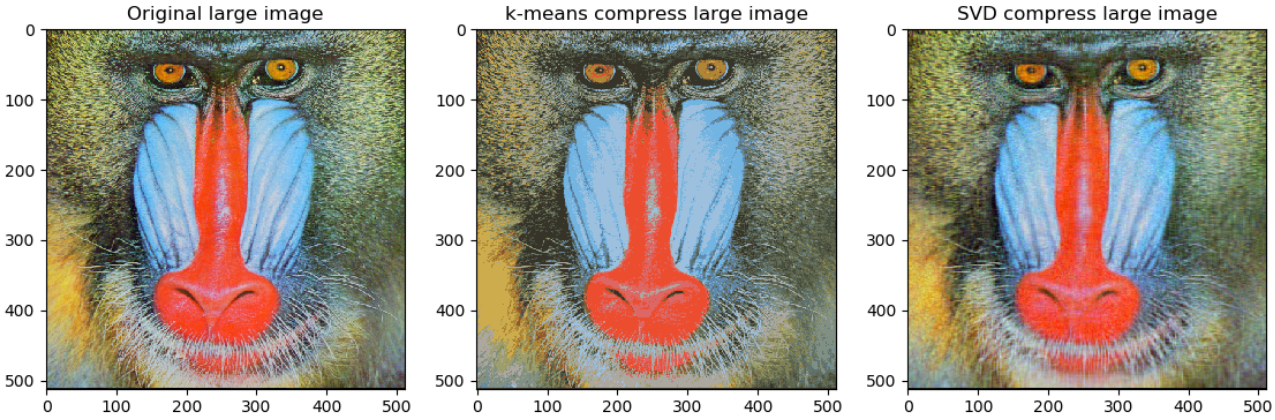

Figure 2.3: This image illustrates the compressed images from the k-means algorithm and SVD. The compressed image obtained from the SVD method uses the first 50 dominant singular values to approximate the original image. We see that the approximated image obtained using SVD is not as accurate or clear as the one obtained using the k-means algorithm.

singular value decomposition (SVD). When the data is given in terms of a matrix, A, of dimensions mxn, the SVD decomposes the data into a sum of r 'basis' matrices; in technical terms $r = \text{rank}(A)$. In particular, SVD allows us to express a given matrix, $A \in \mathbb{R}^{mxn}$ as:

$$A = U\Sigma V^T = \sum_{i=0}^{r} u_i \sigma_i v_i^T, \tag{1}$$

where $U \in \mathbb{R}^{mxm}, \Sigma \in \mathbb{R}^{mxn}, V \in \mathbb{R}^{nxn}$. The vectors u_i and v_i are the components of the matrices U and V, respectively and the quantities σ_i are the known as the singular values. From this, SVD decomposes a given matrix into the sum of its singular components. From this decomposition, we may choose to approximate the matrix A by choosing the first K dominant singular values. In this example, there are 512 singular values, from which we choose the first 50 dominant ones to approximate the large image. The approximate obtained from SVD compares poorly against the output of the k-means algorithm. The explanation for this is due to the fact that singular values are all comparable of about 10^3 orders. Consequently, most of the singular components are required to give an accurate approximation.