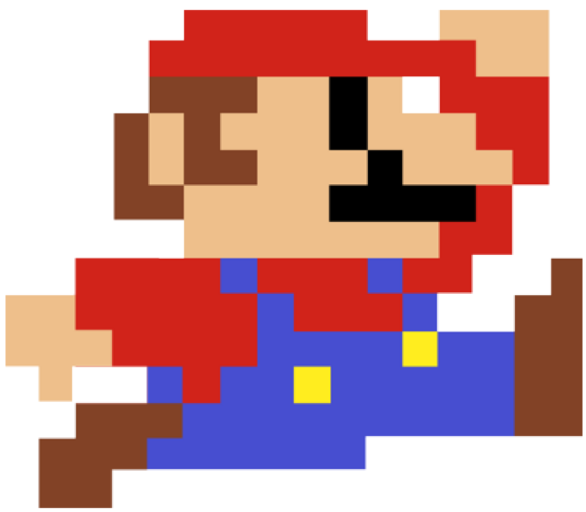
MAT245 Lab 3

Oct 2, 2017

Working with images

Goal: Load and display an image using `scipy`.

A computer screen is a grid of small cells called *pixels*. Images can be represented by shading or colouring these pixels individually. Images on a screen with very few pixels might look like:



As a result, computers represent images using arrays of numbers. A black and white image can be described by an $n \times m$ array $B^{\texttt{img}}$ (ie. a matrix) of integers $0 \leq n \leq 255$. If $B^{\texttt{img}}_{ij} = 0$ then the corresponding pixel is black; if $B^{\texttt{img}}_{ij} = 1$ then it's white. Intermediate values produce progressively lighter shades of grey.

The simplest way to represent a colour image is to use an $n \times m \times 3$ array $C^{\texttt{img}}$. In other words, $C^{\texttt{img}}$ is an array of three $n \times m$ matrices. The first, second and third matrices describe the intensity of red (R), green (G), and blue (B) in any given pixel. (Recall any colour is a sum of RGB values, the primary colors). Again the entries of $C^{\texttt{img}}$ are integers $0 \leq n \leq 255$.

The sort of colour encoding described above produces RGB images. There are other encodings, such as RBGA which includes a transparency channel as well. We won't focus on those here.

Let's try loading an image using `python`'s `scipy` library.

1. Save a colour image from the internet to your working directory.
2. Import `scipy.misc`.
3. Use `scipy.misc.imread` to load your image, both using the RBG scheme and in black-and-white.
4. Use `plt.imshow` to display these images to the screen.

Singular Value Decomposition

Goals:

1. Plot the singular value decompositions of image matrices from the previous section.
2. Compute the rank n SVD approximation a matrix.
3. Compute and plot the relative error for SVD approximations to a matrix.

Background

Recall that a singular value decomposition expresses a $n \times m$ matrix A as a product

$$A = U\Omega V^T.$$

Here:

- U is an orthogonal $n \times n$ matrix;

- V is an orthogonal $m \times m$ matrix;
- Ω is an $n \times m$ matrix of zeros, other than entries $s_1, \ldots, s_k$ on the diagonal for $k = \min(n, m)$.

The numbers $s_1, \ldots, s_k$ are the singular values of A.

Suppose δ_{ij} is an $n \times m$ matrix of all zeros except for the ij^{th} entry, which is 1. Set

$$\Omega(a, b) = \sum_{i=a}^{b} s_i \delta_{ii}$$

The matrix

$$A_r := U\Omega(1, r)V$$

is called the rank r approximation of A. We will see that A_r gives a "pretty good" approximation of A, even for relatively small values of r. This makes SVD approximation a useful tool for dimensionality reduction.

To determine how "good" the approximation is, we can investigate

$$E_r := A - A_r = U\Omega(r + 1, k)V.$$

The Frobenius norm of E_r has a particularly simple form:

$$|E_r|^2 = \sum_{i=r+1}^{k} s_i^2.$$

Particularly useful is the relative error function can then be defined by

$$e(r) := \frac{|E_r|}{|A|} = \sqrt{\frac{\sum_{i=r+1}^{k} s_i^2}{\sum_{i=1}^{k} s_i^2}}$$

Steps: (Part 1)

- The function `numpy.linalg.svd` computes the singular value decomposition of a given matrix (see the documentation). Use this function to obtain a list of the singular values of the black-and-white image matrix $B^{\texttt{img}}$ from the first section.

- If s_n denotes the n^{th} singular value, use `matplotlib`'s `plot` to graph the function $n \mapsto s_n$. Pay close attention to the shape of this graph.

Steps: (Part 2)

- Write a `python` function that takes a matrix and an integer r representing rank, and returns $B_r^{\texttt{img}}$, the rank r approximation to $B^{\texttt{img}}$.
- The functions `numpy.zeros` and `numpy.diag` might come in handy here.
- Use the `scipy.misc.imshow` to display the images represented by $B_r^{\texttt{img}}$ for $r = 1, 10, 25, 100$, and any other values you'd like.

Steps: (Part 3)

- Write a `python` function that takes a list of the singular values of a matrix, a rank parameter r, and returns the relative error $e(r)$.
- Use `matplotlib` to plot the graph of the relative error function $r \mapsto e(r)$ constructed above for $0 \leq r \leq k$.
- Determine the smallest r that can produce an approximation with an error < 0.05.

Questions

Consider the following questions:

1. What is the purpose of using SVD? What value does it give us?
2. How does SVD stack up to other techniques of dimension reduction, such as PCA?

Additional Resources

- Consult Mining of Massive Datasets Section 11.3 (Leskovec, Rajarman, Ullman) for an in-depth explanation of SVD as applied to the dimension reduction of datasets.

- The Netflix Prize was an open data science competition in collaborative filtering to best predict user ratings for film. The best model (which won 1 million USD) used a variant of SVD, SVD++, with Restricted Boltzmann Machines. Learn more about how they did it:

  http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/
  http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf