CENG 499 – Introduction to Machine Learning

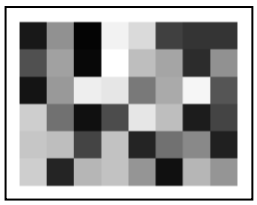
Spring 2017 – Homework 3

Honoring Escher & Bob Ross: The GA Painter

Selim Temizer

Feedback : Informal (no set date), you may post questions on the newsgroup
Due date : June 15th, 2017 (Submission through COW by 23:55)

In GNU Octave, you may easily create a matrix of size $\boldsymbol{m \times n}$ with random real numbers in the range]0,1[with the command `rand(m,n)`. It is also possible to visualize this matrix as a grayscale image with the command `imshow`, or save it with `imwrite` commands. So, for example, the command sequence: `I = rand(6,8) ; imshow(I) ;` will generate and display an image like the one shown on the right. We can also convert a matrix into a column vector by stacking each column on top of each other, and combining a long sequence of columns into a big single column. Therefore, in this homework assignment, we will use column vectors of size $\boldsymbol{s}$ in order to represent grayscale images of size $\boldsymbol{m \times n = s}$.



The first goal of this homework assignment is to complete the missing parts of a Genetic Algorithm implementation in GNU Octave, in which the aim is to work with populations of column vectors (each representing a grayscale image), and trying to evolve the images until we obtain a certain target image. The target image is not given to you, but a fitness function is available that returns a fitness value representing how close an image instance is to the target image (the higher the fitness, the closer the image is to the target image). Furthermore, if you are working with images of size $\boldsymbol{m \times n = s}$, it is guaranteed that the maximum fitness that can be achieved (i.e., the fitness of the target image) is going to be $\boldsymbol{s}$.

The second goal is to run the completed code in order to evolve images of size 5×5, 8×8, 10×10, 15×15, and if your implementation is fast enough, 20×20 or even higher resolution square images. When running your code, you are free to tune the population size, mutation probability and stopping criteria as you wish. Just note that, if you stop training earlier, or if you accept a lower fitness instance as the output of your run, the resulting image will look less like the target image that we are interested in reconstructing.

Note that each time you run the code, it finishes training (according to your stopping criteria), and then generates an image file with the output, and also generates another image file with the plot of the fitness values across the populations. You should save those images and plots, and create a small report in which you provide the parameters and resulting images for all different resolution experiments.

The last optional (bonus) goal is to correctly guess what the target image is, after inspecting the various resolutions of it by your experiments defined in the previous paragraph.

The low resolution images may not be helpful in guessing what the target image is, but if you could get very high fitness images with higher resolutions, it might be easier to guess the target. Also, when looking at generated images, try to squint a little bit, and this might also help interpret the low resolution images, as well.

What to submit? (Use ***only ASCII characters*** when naming your files and folders)

1. Your filled in, and ready-to-be-tested, GNU Octave script file that has the completed Genetic Algorithm implementation.
2. A Word or PDF report that has information about the parameters and the generated plots and images for each run of your script for the different resolution experiments. If you have a guess about the target image, also write it down in your report.

Zip the 2 files, (tar also works, but I prefer Windows zip format if possible), name the compressed file as <ID>_<FullNameSurname> (with the correct extension of .zip or .tar) and submit it through COW. For example:

e1234567_SelimTemizer.zip

Late submissions will not be accepted, therefore, try to have at least a working baseline system submitted on COW by the deadline. Good luck.