Assignment 1: Search
EECS 492: Artificial Intelligence
Fall 2015

Now: Thursday, September 10, 9:00 am
Due: Tuesday, September 29, 11:00 pm

In this assignment, you will search within a very large state-space to find a simple approximation to a given digital image, using a genetic algorithm (sect.4.1.4 in Russell & Norvig (R+N)).

Look at `http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/`. Follow his links to get more details, and to download his code. It's worth reading the comments on these and related pages. Search in YouTube, for "evolution Mona Lisa".

Roger Alsing's algorithm implements a kind of gradient ascent ("hill-climbing") along a single path. Alsing's code generates a single random descendent of the current specification for the triangles approximating the image. If the "child" is superior to the "parent" (in terms of distance to the target image), then the child replaces the parent. Otherwise, the child is discarded. Then the loop continues.

You will implement a more general and sophisticated genetic algorithm, and you will compare its performance for different parameters.

Genetic Algorithm Search for Image Approximation

The state space for the search is the space of images (i.e., arrays of colored pixels). The goal state is specified by a given target image. An individual state in this search has a "genome", which specifies how its image is generated from a finite set of basis shapes (e.g., triangles, polygons, circles, etc.) with given color and transparency.

To compute the fitness $F_t(s)$ of a state s with respect to a target state t, start by computing the pixel-wise distance between s and the target state t. Since each pixel has (R, G, B) components, we can consider a color image with n pixels as a vector with $3n$ elements. The Euclidean distance between s and t is $|s - t|$. By the triangle inequality, we know that $|s - t| \leq |s| + |t|$. Therefore, we define fitness to increase without bound as s gets very close to t.

$$F_t(s) = -\log \frac{|s - t|}{|s| + |t|}$$

Since the quotient must be in $[0, 1]$, we know that $F_t(s)$ is in $[0, +\infty]$.

The genetic algorithm has a population of N states in each evolution step. K pairs of parent states are selected to have a child, with probability of being selected proportional to fitness. A state may be selected as a parent zero, one, or multiple times in a given generation. Figure 4.6 illustrates how the genome of a child is derived from the genomes of the parents.

Adding the children gives a population of $N + K$ states. Compute their fitnesses and keep the top N as the next generation. (In this framework, Roger Alsing's algorithm has $N = K = 1$ and the child has two copies of the same parent.)

Details on Inheritance, Cross-over, and Mutation

The genome for these individuals consists of an ordered list of P polygons, each of which consists of an ordered list of the coordinates of its vertices and the values for the red, green, blue, and alpha components of its color. In short, the alpha component indicates the opacity of a color, with 1.0 meaning completely opaque and 0.0 meaning transparent. C++ and Python have APIs that provide ways to draw shapes with specified values of red, green, blue, and alpha, and we will provide code that calls these library functions. You can refer to `http://en.wikipedia.org/wiki/Alpha_compositing` for more information. Python, for example, has a nice and lightweight library called Pygame that can process images.

The list of polygons is ordered because the order in which the polygons are drawn is significant. For example, drawing an opaque polygon will obscure the polygons that have been drawn previously.

The individuals at generation 0 are generated randomly, with each individual consisting of P triangles with random vertices and colors.

For simplicity, we make the number of polygons $P = 100$ constant for the population. After two parents are selected, we perform cross-over by picking a random cross-over point C between 0 and P. The child inherits the first C polygons from one parent and the last $P - C$ polygons from the other parent. Note that when $N = 1$, each child has two identical parents, and cross-over has no effect.

After cross-over, a child can be mutated by having its genome altered slightly in a random way. Here is a list of possible mutations:

(1) Replace a polygon with a random new polygon of the same number of sides.
(2) Swap the positions of two polygons in its ordered list.
(3) Slightly alter a polygon's color.
(4) Slightly alter the position of a polygon's vertices.
(5) Add a vertex to a polygon
(6) Remove a vertex from a polygon

Hint: If these mutations are very large with high probability, most children will be deformed and die (i.e., their fitness will be less than their parents). If these mutations are very small with high probability, the children will be more likely to live, but fitness may increase quite slowly. Expect a certain amount of tweaking of the mutation rates and amounts of mutation to help the program produce good results in a reasonable amount of time. (But see one of the Further Investigation topics.)

You will quickly notice that computation time is dominated by the fitness measure. To compute the fitness of a state, you must render its image from its genome, and then compute the pixel-wise distance between that image and the target image. Hint: Computation of the fitness measure depends on the size of the image, so do your algorithm development and debugging with small images.

Evaluating search performance

Here are two possible ways to evaluate the algorithm's performance:

1. The total number of states created (and the total number of times the fitness function must be computed) to reach a certain tolerance of distance from the target image. If it takes T generations after the initial population N, the search effort will be $E = N + T * K$.

2. The fitness (or distance to target) reached after a certain number of states have been generated. We will use this metric in our assignment.

Provided code

We will provide code that uses library functions for:

reading the image from a file,
rendering a polygon onto an image buffer,
computing the Euclidean distance of pixel colors between two images, and
writing images to an animated PNG (APNG) file.

All of these are library functions in Pygame (for Python programmers) except for the APNG routines. "pngmerge", a third-party library for APNG routines, will be provided.

pngmerge is also available at `http://kichiki.web.fc2.com/pngmerge/`.

In C++, there are good image processing libraries such as OpenGL, OpenCV, and OpenImageIO. We're in the process of selecting library functions for the above tasks in C++. In the meantime, please feel free to explore on your own.

Assignment:

(1) Implement this genetic search algorithm for image approximation, parameterized by P, N, K and E. Emphasize clarity, modularity, and modifiability in your code.

Select five target images of different sizes and contents. Two should be the portraits of Mona Lisa and Charles Darwin. The other three are of your selection.

(2) Tune the parameters N and K. Set P=100. Evaluate the fitness reached after search effort $N + TK = 25,000$, for all values of N and K from $\{1, 2, 4, 8\}$. Provide a graph, showing fitness achieved as a function of (N, K). You may display fitness as a function of (N, K) as a 3D terrain plot, as a color-coded 2D plot, or any other way that conveys the information clearly. For each of your five images, what is the best value for (N, K)? What is the best overall value for (N, K)?

What mutations do you allow in your algorithm? We have provided six possible mutations before, but you may choose not to use all of them or add others.

What is your method for selecting parents? One option is to let the probability of being selected proportional to fitness. You are free to use others.

(3) Create an HTML page to demonstrate your genetic algorithm search. For each target image, show the image, an APNG animation showing the evolution of the highest-fitness state, and a graph of its fitness as a function of generation T.

An example is provided at `http://umich.edu/~schapel/492/A1results.html`. (It appears that Firefox is necessary to view the APNG animations properly.) The animation should start with the initial random state, and then display the highest-fitness image after every 1000 children have been created, at 15 frames per second. For example, if $K = 10$, this will give a frame every 100 generations, or 1500 generations per second. This is intended to make your demonstration animations relatively brief. Display a counter for the generation number. You may want to modify these parameters for your own debugging purposes.

What to submit:

A zipped directory with two subdirectories: html and code. Name your directory "A1-*unique*", where "*unique*" is replaced by your own uniquename. For example, since my uniquename is kuipers, I would name this directory A1-kuipers.

The html subdirectory contains your HTML page with associated image files. Your HTML page should have the following contents:

– Images and animations related to Assignment (1)
– The graph of Assignment (2)
– Answers to Assignment (2)

The `code` subdirectory contains the C++/Python code for your program. Please provide a README file with enough information to run your code. Specifically, your program should take 3 parameters, N, K, E, where N, K are as defined previously, and E is the search effort, defined as $E = N + K * T$ (T is number of generations). Your program should terminate after T generations of evolution and output the highest fitness value in the last generation.

Your source code must compile and run under g++ 4.6 or Python 2.7 on a specified image file, for specified values of the parameters N, K, and E.

Submit on CTools, through the Assignments tool. If for any reason that doesn't work, submit it to the Drop Box, and send email.

Scoring details

(20 points) Code compiles and runs.
(30 points) = 5 × [animation (4 points) + fitness curve (2 points)]
(30 points) = 6 × [fitness(N,K) plots (3 points) + best values (2 points)]
(5 points) Answer to mutation question
(5 points) Answer to parent selection question
(10 points) Overall clarity and quality, of both code and presentation.

Extra credit (amount to be determined on a case-by-case basis) for impressive results on one or more of the following additional questions.

For further investigation:

Here are some interesting questions you may want to investigate, once you have your assignment done. They are optional. If you choose to do them, put your answers to each question in a separate file. Name the file by the label of corresponding question, e.g. *a.txt*, *b.html*. If your work has image to present, use an html file, otherwise you can use txt.

(a) Are there better choices for a distance metric than ones based on the Euclidean norm $||x||_2 = [\sum_i x_i^2]^{1/2}$, such as $||x||_1 = \sum_i |x_i|$ (the l_1 norm), or $||x||_\infty = \max_i |x_i|$?

(b) Suppose your target image is a large, high-resolution image. Since it is easier to compute the fitness measure for small low-resolution images, can you define a more efficient hierarchical search using a "pyramid" of lower-resolution versions of the original target image? Find a good approximation to the lowest-resolution image, then use that image to initialize the search at the next higher-resolution level of the pyramid, and so on.

(c) Compare different basis sets of shapes: triangles, polygons, ellipses, etc. How much difference does the choice of basis make?

(d) Try starting with two "Adam" and "Eve" states and let the population grow, culling low-fitness states so the population is no greater than N at the beginning of each generation.

(e) Can you measure the "diversity" of the population of states? Does diversity increase or decrease with time? How important is it? (The research literature in genetic algorithms suggests that decreased diversity is important for effective search. Can you observe this?)

(f) *Is sex a good idea?* In asexual reproduction, there is only one parent, and mutation is the only source of change. In sexual reproduction, the genomes of the two parents cross over before mutation takes place, which is an additional source of change from one generation to the next.

You should be able to compare the speed of convergence with and without sexual reproduction in your model. Actually, there are two models for asexual reproduction that should be compared. Suppose our base case is denoted $sexual(N, K)$, meaning that we have sexual reproduction with a population of N and K children each generation (from $2K$ parents, not necessarily distinct). The obvious comparison is with $asexual(N, K)$, with no cross-over and only K parents. However, one might argue that this under-samples the population for parents at each generation, so we also compare with $asexual(N, 2K)$.

Prior to running the experiment, what is your hypothesis about the relative speeds of convergence of the three reproduction modes? What is your justification for that prediction?

Then, run the experiment, and see how this changes your conclusion.

(g) Adaptive simulation. We have discussed the observation that the result will depend on the mutation rates and amounts. Rather than tuning those rates by hand, can we use the evolutionary process to identify good rates.

Suppose we start with a certain choice of values for the parameters defining those mutation rates and amounts. Instead of keeping this parameters as global variables, we will associate them with each individual in the population, so they become part of the genome. When a child is created, it inherits these values from its parent(s), corrupted with a certain amount of Gaussian noise. (For the initial state, the individuals are randomly perturbed from the given values.)

My hypothesis is that these parameters, like the others in the search, will converge to the best values.