6/13/2021

MET CS 767 Assignment 5T: GA's

Kimberly Nestor

Please fulfill the requirements in the gray text, handing in this filled-out template and the accompanying source code, leaving the gray text and the headings unchanged.

You are to implement the *traveling salesman problem* using a genetic algorithm in the manner outlined below, particularly in the *Crossover* section—which you are advised to read first. You can assume that there is a route connecting every pair of cities, and that no two distances are equal.

Eric Braude created this technique independently as a way to perform smooth crossover for Traveling Salesman but it may be published. If so, please do not use the work of others in responding.

1. Representation of the Data

Explain how the application will represent *the data*. Include how the following example is to be represented: B(oston) to L(ondon) 3(k miles), L to M(umbai) 4.5, M to S(hanghai) 3.1, S to L 5.7, B to M 7.6, and B to S 7.8.

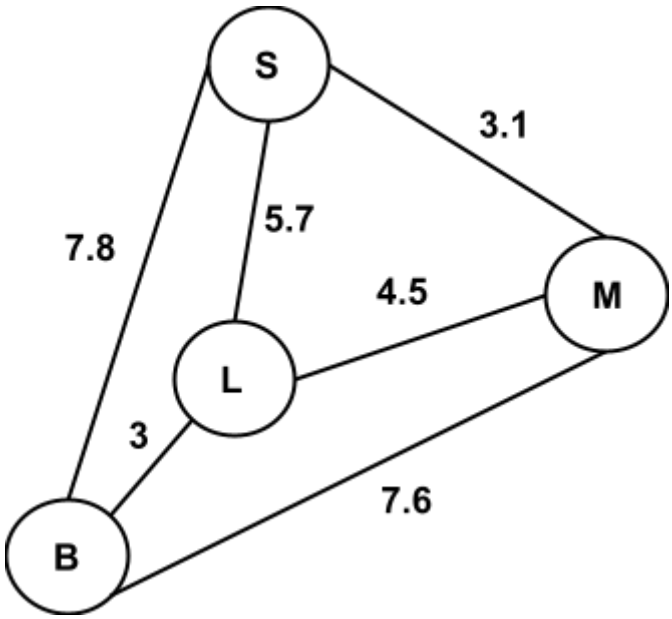

Figure 1: graph representation of city data for the travelling salesman problem

	B	S	L	M
B	0	7.8	3	7.6
S	7.8	0	5.7	3.1
L	3	5.7	0	4.5
M	7.6	3.1	4.5	0

Table 1: matrix representation of data for the travelling salesman problem

I decided to think of the application of the travelling salesman problem in a graph context. In the first image above I diagrammed a weighted graph with each city visited by the salesman as the nodes and the distances between each city as weights on the edges. Though this helps with a visual representation of the data provided above, the practical implementation of this data will be based on the subsequent table, which represents the data provided in the graph. For this problem I will represent the data as a matrix of the visited cities and each intersection of the matrix represents the distance travelled by the salesman to get from one city to another.

2. Representation of a Route

Explain how the application will represent *a route* starting and ending in Boston. Include how the following route is to be represented: B to L to M to S to B.

	Start city B	First distance city	Second distance city	Third distance city	End city B
Numerical representation	0	1	2	3	0

Table 2: numerical representation of genetic data, ranking of travel city options by distance

In order to implement the crossover method in the genetic algorithm, I converted the choices in the cities the salesman can visit into a numerical representation. The value 0 will always represent city B as it is both the start and end city. The value 1, out of the three city options to visit, represents the city with the least travel distance. Whereas 2 represents the city with the

second smallest travel distance out of the three options and 3 represents the city with the largest travel distance.

Using the example from the Crossover Question 3, the following would be an example representation of how the above numerical system would work in the genetic algorithm. The crossover point in the example is at position 1.

Parent 1: 0 2 2 1 0

Parent 2: 0 1 1 1 0

Child: 0 2 1 1 0

Using the numerical representation of the genetic data presented in Table 2, the route taken by the crossover produced by child 1 (02110) would follow the node pattern:

B → M → S → L → B.

The salesman would start out at node B, then select the city with the second closest distance node M, then the city with the first distance S. At which point it would attempt again to select the first distance which would be node M, however this node has already been visited so the algorithm would select node L and then finally return to node B.

3. Crossover

Provide a crossover function. The implementation should create a child from two parents as in the following example. Bold and -italics are added to clarify what part of the child comes from what part of the parent.

Parent 1 **Boston 2nd closest unvisited city[1] 2rd closest unvisited city[2] closest unvisited city Boston**

Parent 2 Boston closest unvisited city closest unvisited city closest unvisited city Boston

Assuming that the crossover point is at 1, the child route will be

Boston 2nd closest unvisited city *closest unvisited city closest unvisited city Boston*

[1] i.e., compared to all other direct hops from the city just visited (Boston in this case)
[2] i.e., compared to all other direct hops from the city just visited

```
[34] import numpy as np
```

```
def crossover(parent1, parent2):
  "This function takes two parent as arrays and produces a child using the \
  single point crossover method."
  co_pt = np.random.randint(1,len(parent1)-1)
  child = parent1[:co_pt] + parent2[co_pt:]
  return(child)

crossover([0, 2, 2, 1, 0], [0, 1, 1, 1, 0])
```

```
[0, 2, 1, 1, 0]
```

4. Mutation

Explain (clearly) how the application will perform mutation.

```
def mutation(child):
  "This function takes a child as input and performs a mutation on one value \
  of the numerical representation. A while loop is used to ensure a mutation \
  has been performed on the returned child."
  mu_loc, mu_num = np.random.randint(1,len(child)-1), \
                   np.random.randint(1,len(child)-1)
  while child[mu_loc]:
    if child[mu_loc] != mu_num:
      child[mu_loc] = mu_num
      return(child)
    else:
      mu_num = np.random.randint(1,len(child)-1)


cross_child = crossover([0, 2, 2, 1, 0], [0, 1, 1, 1, 0])
print("cross_child = ", cross_child)
print("mut_child = ", mutation(cross_child))
```

```
cross_child =  [0, 1, 1, 1, 0]
mut_child =  [0, 3, 1, 1, 0]
```

The application will perform mutation by first randomly selecting a location in the numerical representation of a route (except for start and end city 0), then replace the value with a random number between 1 and 3 (non inclusive) [1, 2]. In the example shown above a child, [0 , 1, 1, 1, 0], is used as input to the mutation function. After mutating the numerical representation of the child is [0, 3, 1, 1, 0]. The second location in the list was changed from 1 to 3, meaning that instead of selecting the path with the least distance the mutated the child will then select the path

with the third shortest distance (the longest distance). I decided only to mutate one value in the numerical representation of a route, because the routes are quite short and mutating more than one value would represent a large change in the initial path of that child. Therefore since there is only a single value mutation in the numerical representation, I added a while loop to ensure the child does in fact undergo a mutation and not a pseudo mutation; where the original value is randomly selected so the resulting mutation is in fact just the original numerical representation [1, 2]. This while loop is particularly important since mutation will only be performed on a probability basis, so selected children should undergo a true mutation, if not then the actual probability of mutation will be much lower than intended.

5. Result on the Given Data

Describe the result from executing on the data … *B(oston) to L(ondon) 3(k miles), L to M(umbai) 4.5, M to S(hanghai) 3.1, S to L 5.7, B to M 7.6, and B to S 7.8.* What do you think of this result? Explain.

```
[822] #import given data
      data = np.squeeze(pd.read_csv(dpath, sep='delimiter', engine='python').values)

      routes_lst = [i.split(',') for i in [i.replace('}', '') for i in \
                                           [i.replace('\\', '') for i in data]]]
      routes_lst = list(filter(all_isdigit, routes_lst))
      routes_lst = [list(map(float, i)) for i in routes_lst] #B,S,L,M

      enum_routes = [list(enumerate(i)) for i in routes_lst]

      #convert to dict
      routes_dict = {i: enum_routes[i] for i in range(len(enum_routes))}

      #run genetic algorithm to find shortest route
      gen_alg(pop_size=25, num_gener=5, mut_prob=0.20, route_data = routes_dict)

      Generation: 0.  Route: [0, 1, 1, 2, 0].  Distance: 18.4
      Generation: 1.  Route: [0, 1, 1, 1, 0].  Distance: 18.4
      Generation: 2.  Route: [0, 1, 1, 1, 0].  Distance: 18.4
      Generation: 3.  Route: [0, 1, 1, 2, 0].  Distance: 18.4
      Generation: 4.  Route: [0, 1, 1, 2, 0].  Distance: 18.4
```

The above image is the result of executing the genetic algorithm I wrote to implement the travelling salesman problem [1, 2]. I ran the algorithm for 5 generations after which it provided a few route options, all of which had a total travel distance of 18.4. Since we have such a short travel route I'm not surprised that there is not alot of variety in the genes/ numerical representation of the data or in the total distances provided [1, 2]. I will say though that I think this is the most accurate result in terms of the least distance travelled, while ensuring each city is visited once and the salesman returns home to Boston. During implementing this algorithm I performed several checks, one of which was to manually calculate various numerical representations the algorithm developed and 18.4 was the lowest distance I came across.

6. Result on Your Data

Describe the result from executing on illustrative data of your choice. What do you think of this result? Explain.

```
#import new data
data = np.squeeze(pd.read_csv(dpath_new, sep='delimiter', engine='python').values)

routes_lst = [i.split(',') for i in [i.replace('}', '') for i in \
                                     [i.replace('\\', '') for i in data]]]
routes_lst = list(filter(all_isdigit, routes_lst))
routes_lst = [list(map(float, i)) for i in routes_lst] #B,S,L,M

enum_routes = [list(enumerate(i)) for i in routes_lst]

#convert to dict
routes_dict = {i: enum_routes[i] for i in range(len(enum_routes))}

#run genetic algorithm to find shortest route
gen_alg(pop_size=25, num_gener=5, mut_prob=0.20, route_data = routes_dict)
```

```
Generation: 0.  Route: [0, 1, 2, 1, 0].  Distance: 20.799999999999997
Generation: 1.  Route: [0, 1, 2, 1, 0].  Distance: 20.799999999999997
Generation: 2.  Route: [0, 1, 2, 1, 0].  Distance: 20.799999999999997
Generation: 3.  Route: [0, 1, 2, 2, 0].  Distance: 20.799999999999997
Generation: 4.  Route: [0, 1, 2, 1, 0].  Distance: 20.799999999999997
```

In the above image I implemented my GA on new data (slightly altered from the original). The algorithm converges pretty quickly to the minimum distance needed to travel through all cities once. It was nice to see that the algorithm could be extrapolated to other types of data. Other positive notes from trying the algorithm on other data is that it was able to come up with different numerical representations in order to achieve the minimum travel distance.

7. Source Code

Paste your source code below. It should accompany this doc as well.

Colab link:

https://colab.research.google.com/drive/1nc5Vl6q1fqOaQgyj8zTyh5M7S3Zg9VSk?usp=sharing

8. Comments on the Design

If possible, compare the performance of this TSP GA with at least one known TSP GA implementations.

Supply what could be improvements on your design and explain.

I sought out quite a few sources so I could feel comfortable that I understood how to implement a GA correctly and specifically with regard to the TSP [3-7]. However I will focus on comparing my GA implementation with reference [8]. One large difference between the implementation in reference [8] and my implementation is that there are alot more helper functions and alot of the functionality of the GA is broken down into separate parts. While I do have a few functions my code is not as streamlined as reference [8]. I haven't run their implementation but breaking down the methods in this way can lead to more efficient code.

I came across this article when I was doing early research to ensure I implemented the GA correctly, and I purposely did not read it until after I implemented the TSP problem. I'm very happy to say that in the selection of the mating pool both algorithms used "fitness proportionate selection" and "Tournament selection" when selecting the mating pool for each generation. I decided to do this type of selection to ensure I was staying true to the laws of evolution and giving my algorithm the best change to converge quickly to the lowest fitness. While a difference in crossing genetics is that I did a single point crossover while reference [8] did multipoint crossover. I don't think this is a huge difference in implementation for my algorithm because my numerical representation is shorter than that in the reference.

For mutation both algorithms implement a probability based mutation, whereas the reference uses swap mutation of cities, I implemented a random change in one number in the gene/numerical representation. Since this essentially amounts to a swap in city, this is another similarity between the two GAs. The reference implemented their GA on a much larger dataset so they were able to obtain a right skewed line graph from their lowest distances obtained by generation, whereas I had far less data and the GA came upon the minimum distance from the first generation.

Improvements:

Though I did implement my algorithm on another dataset, I did not change the graph representation. I generally try to program so that my code can be extrapolated to all sorts of data. However this is my first time implementing a genetic algorithm and only my second time traversing a graph, so I would be interested to see whether my algorithm can work effectively on large graphs. Other potential shortfall of my algorithm when increasing graph complexity, is I'm not sure whether it will be efficient in computational complexity as I used quite a few loops in the algorithm and these tend to increase running time.

References

Show that you used a wide variety of resources by listing them below and clearly indicating in the body above where you used. Make sure to use proper referencing in your paper. We suggest using APA format, but other formats are fine as long as they clearly distinguish your work from work of others in your response. In general, observe the stated plagiarism rules.

[1] Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (Introduction to Artificial Neural Networks with Keras: pp. 279-295). O'Reilly.

[2] Marsland, S. (2015). *Machine Learning: An Algorithmic Perspective* (Preliminaries: pp. 15-20, Neurons, Neural Networks, and Linear Discriminants: pp. 43-49, Unsupervised Learning: 281-287). CRC Press.

[3] Brownlee, J. (March, 2021). Single Genetic Algorithm from scratch in Python. Machine Learning Mastery.

https://machinelearningmastery.com/simple-genetic-algorithm-from-scratch-in-python/

[4] Mallawaarachchi, J. (July, 2017). Introduction to Genetic Algorithms - Including Example Code. Towards Data Science.

https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3#:~:text=A%20genetic%20algorithm%20is%20a,offspring%20of%20the%20next%20generation.

[5] Gad, A. (July, 2018). Introduction to Optimization with Genetic Algorithm. Towards Data Science.
https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b

[6] Tutorials with Gary. (2020, May 24). *Convergence - Writing a Genetic Algorithm from Scratch* [Video]. YouTube. https://www.youtube.com/watch?v=y8Tm4hlbLCE

[7] n Sanity. (2013, Aug 18). *Travelling Salesman Problem* [Video]. YouTube.

https://www.youtube.com/watch?v=SC5CX8drAtU&t=1s

[8] Stolz, E. (July, 2018). Evolution of a salesman: A complete genetic algorithm tutorial for Python. Towards Data Science.

https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35

Evaluation

	D	C-	C+	B-	B+	A	Letter Grade	%
Clarity	Disorganized or hard-to-understand		Satisfactory but some parts of the submission are disorganized or hard to understand	Generally organized and clear	Very clear, organized and persuasive presentation of ideas and designs	Exceptionally clear, organized and persuasive presentation of ideas and designs		0.0
Technical Soundness	Little understanding of, or insight into material technically		Some understanding of material technically	Overall understanding of much material technically	Very good overall understanding of technical material, with some real depth	Excellent, deep understanding of technical material and its inter-relationships		0.0
Thorough-ness & Coverage	Hardly covers any of the major relevant issues		Covers some of the major relevant issues	Reasonable coverage of the major relevant areas	Thorough coverage of almost all of the major relevant issues	Exceptionally thorough coverage of all major relevant issues		0.0
Relevance	Mostly unfocused	Focus is off topic or on insubstantial or secondary	Only some of the content is meaningful and on topic	Most or all of the content is reasonably meaningful and on-topic	All of the content is reasonably meaningful and on-topic	All of the content is entirely relevant and meaningful		0.0
						Assignment Grade:		0.0

The resulting grade is the average of these, using A+=97, A=95, A-=90, B+=87, B=85, B-=80 etc.

To obtain an A grade for the course, your weighted average should be >93. A-:>=90. B+:>=87. B:>83. B-:>=80 etc.