Machine Learning Engineer Nanodegree

Capstone Proposal

Frank Corrigan
April 24, 2017

Domain Background

Approximately 17 million individuals in the US (according to 2015 data from runningUSA.org) identify themselves as runners. These are people that have participated, at least once, in an organized running race. This cohort is continuously searching for recommendations on how to run faster, who to run with, and where to find more enjoyable runs. That final question is the topic of focus in this capstone project. A particularly common method of deciding where to run is to ask other runners. Recently, websites such as Strava or MapMyRun have made the answer to this question more shareable. Any runner can throw a GPS-enabled watch on their arm, go for a run, upload the data to their site of choice, and other runners and view what routes other runners are running most often. The most popular routes are regarded as the best routes to run.

However, not every runner enjoys the same scenery or can take the time to commute to those best run locations. It would be advantageous if we could find the best or optimal run according to personal taste directly from where we are staying or living. Since time is finite and exploration of the options for running routes seems infinite, machine learning can assist in determining these optimal paths. The problem of considering which route to take is not unique to runners and running paths - optimal path finding problems exist and have been studied in depth in a variety of fields including naval ship ocean traversal, automobile navigation through road networks, logistics (recall the infamous traveling salesman problem), and even in a related activity, cycling. An optimal path, depending upon its application, could mean the shortest path, the most fuel efficient path, or the fastest path. Each of these fields lends information for how to think about determining the best running route from an individual's current location.

Leigh M. Chinitz had a similar idea many years ago. In 2004, he registered a US patent on an algorithm that created closed loops for cyclers. The website that leverages this algorithm, RouteLoops.com, asks the user for a starting location and target mileage and creates a looped route the cycler (or walker) can follow. Additionally, there are some high level preferences you can select such as road selection type. This research is helpful in understanding a good method for creating loops from the same start and end location. However, the RouteLoops algorithm is explicit and doesn't incorporate preferences such as "run past water" or "go past historical monuments." I should also note that MapMyRun.com also has a demo product that creates loops for runners, but in it's current form (April 2017) it only works in Paris, France You can upgrade to use it in other cities.

On a personal note, I've been a competitive runner for many years through high school, college, and ever since. When I moved up to Melrose, MA -- 10 miles outside Boston -- a few months ago I was delighted and overwhelmed by the numerous options of trails I could explore. I do love exploration, but

as I worked through Udacity's Smartcab Reinforcement Learning project I began to realize that running a new trail system and finding the best loops is very similar to the q-learning algorithm I was creating to teach a car to drive. I wondered if my computer could help me find the best trails in a much timelier fashion than just running every day. From years of experience, I have good intuition for what constitutes a good trail or a good run. If I can map a reward system to attributes of the trail system, we can use simulation and reinforcement learning to determine the best routes even before lacing up our sneakers.

--http://www.runningusa.org/statistics
--http://users.iems.northwestern.edu/~dolira/dolinskaya_dissertation.pdf
--https://en.wikipedia.org/wiki/Travelling_salesman_problem
--http://patft.uspto.gov/netacgi/nph-Parser?u=%2Fnetahtml%2Fsrchnum.htm&Sect1=PTO1&Sect2=HITOFF&p=1&r=1&l=50&f=G&d=PALL&s1=7162363.PN.&OS=PN/7162363&RS=PN/7162363

Problem Statement

The problem statement in it's simplest question form is "tell me the best (or optimal) running route from my current location." A runner wants to run the best runs as soon as possible, but it takes time to learn where they are and what your surroundings have to offer. Where are the trails or bike paths that will allow me to enjoy nature or avoid intersections and getting hit by cars? Where are the enormous uphills that I'd like to avoid today? Are there historical structures are tourist attractions that I'd like to run past and see in this area? Is the neighborhood safe for me to run through? Are there restrooms or water fountains I can pass along the way? Each one of these questions, in total, is a piece of this problem.

A 10 mile run could have dozens of different split point decisions that need to be made. If a runner ran a trail system 1000 times, they'd be able to pinpoint the most rewarding runs given the characteristics of that area. However, 1000 runs could take several years! The goal is to speed up this learning process for each individual runner. Applying reinforcement learning this problem will allow the runner to know the optimal decision at each split point in order to maximize the utility of a running loop based on a desired target mileage number. We can measure that by A) plotting on a map what the algorithm suggests as the best path and B) inspecting each decision at different points with knowledge of what lies ahead in that best path. If this works for me -- in the MIddlesex Fells Reservation -- it should work anywhere we can obtain map data for.

Datasets and Inputs

The dataset for this project will come from OpenStreetMaps.org. This open source geospatial data can be downloaded in .OSM format, which is very similar to XML. Further, there are many good resources on how to extract and manipulate these files. The map for this project will include parts of Malden, Melrose, and Winchester, MA. which is primarily the Middlesex Fells Reservation area. Technically, the bounds include <bounds minlat="42.4251000" minlon="-71.1318000" maxlat="42.4702000" maxlon="-71.0699000"/>.

42.4436421, -71.0721040

This data includes both nodes and ways. Nodes -- which include latitude and longitude coordinates -- identify points on a map and ways -- which include collections of nodes -- identify links between nodes.

Nodes and ways are tagged with different types of identifiers. For example, a node may be tagged with v="traffic_signal" meaning there is a traffic light at those coordinates. Waypoints have significantly more tags. One very common tag is k="name" usually accompanied by a street name. The starting location for this project will be Stone Place in Melrose, MA - where I currently reside. Three other tags that will be leveraged to assign rewards to different ways are

1) "highway" equals "path" or "footpath" which will indicate that the way is a running path prefered over the road.

2) "tourism" equals "alpine_hut", "wilderness_hut", or "viewpoint" which indicates either natural-setting shelter (usually in cool places) or a lookout where you can see something neat (in this case it will be a skyline view of Boston). This tag also includes "artwork" (it's always neat to run by a Banksy).

3) "historic" equals anything usually means something cool. Count it.

3) "natural" equals anything. There are a variety of tags to tell us about the way here including "peak" (is it a mountain peak) and "water" or "spring" (does it go past a body of water).

Each of these tags will provide us with information about the reward system of a particular course. At a high level (more details in the solution section), the algorithm will parse this data, move from node to node along the ways, increment its knowledge base (q-table) of the best ways based on the tags, and create a policy for each decision point along the route. Of course, this is in addition to determining a proper target mileage loop for the runner.

--http://www.openstreetmap.org
--http://www.openstreetmap.org/#map=13/42.4530/-71.1269
--http://wiki.openstreetmap.org/wiki/Hiking

Solution Statement

The optimal route -- for this particular solution -- will create a loop (start and end at same location) and maximize miles on "trails" (which can include hiking trails, running paths, or bike paths) rather than roads, include picturesque water views, and incorporate the highest number of historical or natural landmarks possible. Balancing other variables will result in alternative solutions to this problem. For instance, if an individual is recovering from a hamstring injury they will want to avoid hills and as such minimize elevation gain to yield the optimal route. This implementation will focus on mileage on trails, aesthetically pleasing or educational surroundings.

The algorithm under development will include 2 user inputs. A starting location and a target mileage for the run. Simultaneously, we'll use a q-learning algorithm to determine the best route for the target mileage. A proof of concept, called proof.py, has been created to show how this solution will work.

Ultimately, you'll have a q-table with q-values for each possible move that will still loop you from the same location and route your run through the best places the area has to offer (+trails, +views, +history).

This entirely answers the question, if I'm at location X on this 5 mile run, which direction should I go next in order to maximize utility? We'll be able to look at the optimal policy the algorithm determines, look at the path and its contents on a map, and determine if the algorithm has learned what we would consider the best policy.

A successful implementation will yield a running route that is within one-quarter mile of the target mileage for the run and will include the maximum number of waterfront paths and historic landmarks given the vicinity of the user's location. This may be challenging, if not impossible, in some cases. For example, the desired start location may be in Nebraska where there is only a single road in two directions for 10+ with no trails, no water, and no historic landmarks in the area. Since success in that instance would not be achievable, this implementation will narrow the geographic boundaries considered for training to a portion of Malden, Melrose, and Winchester, Massachusetts the majority of which is called the Middlesex Fells Reservation. A successful algorithm will have the ability to render the best (optimal) running route from most any location and should be tested as such.

Benchmark Model

SInce no single source will serve as a benchmark of the output of this model, I will leverage two existing tools; RouteLoops.com and Strava.com. RouteLoops will show me how quickly the algorithm needs to perform. While it doesn't have to be as fast as RouteLoops, it will have to be within "range" of the RouteLoop builder. RouteLoops can build random loops within seconds and this algorithm should perform similarly while providing a more personalized, optimal route. Additionally, we can compare quality of decisions the algorithm makes by observing decisions RouteLoop makes. Did we create a loop or was it an out and back? Was the loop a figure eight? Did our algorithm select more trail intensive loop than RouteLoops?

Second, Strava.com will show you the most competitive segments. Segments are not loops, they are pieces of loops. For a given area we can identify the most competitive segments (perhaps using the Strava API) and see if the 'optimal' running path runs along any of these segments.

Evaluation Metrics

Until we have a computer-human feedback loop in place where the algorithm suggests a loop, the runner runs it, and they rate how awesome the route was we won't have the perfect data to measure (and improve) the success of the algorithm. Instead, to start, we'll consider 2 criteria for the success of the algorithm.

First, time taken to compute the optimal path. If this exceeds 30 seconds, we have failed. I don't even want to wait more than 30 seconds for my Garmin (GPS watch) to find a satellite connection and most runners won't want to wait more than a few seconds for a suggested run... they'll just go. The second evaluation metric is the quality of the run. For this, I will use myself and another runner who is familiar with the Fells and we will both be able to give a subjective assessment of how good the loop is relative to other routes we know we could take. Despite its unscientific nature, this will give a better assessment of the quality of the suggested run than anything else available.

Project Design

The steps required to achieve this solution:

Build proof of concept to see if reinforcement learning can be applied to creating a loop on a grid with a randomly assigned reward system (already done). This includes understanding the reward trade-off between path characteristics and learning how to loop. It also includes understanding the impact of adjusting algorithm parameters (alpha, gamma, epsilon). The proof of concept can be found here: https://github.com/FrankRuns/machine-learning/blob/master/projects/capstone/proof.py

Retrieve data from OpenStreetMap.org for the grid where testing of the algorithm will occur -- this is a section north of Boston described in the "Datasets and Inputs" section above (also already done).

Set up API that the program can call that gives the distance (as the crow flies) between the start location and each point that is visited on the loop. This will help the algorithm learn the relationship between selecting the "closest" point to the start / end location and the mileage elapsed. Additionally, we'll need to use this API to keep track of the miles run through each iteration. We could try using the OSRM server, but I have a feeling we'll reach our query limit too quickly.
https://github.com/Project-OSRM/osrm-backend/blob/master/docs/http.md

Define logic to map path characteristic (trail, historic site, water view, or viewpoint) to a reward for each path that is visited on the loop. The challenge here will be balancing the reward system to learn both A) how to build a loop and B) maximize the best path characteristics possible from that start location

Create NoSql database that will store metadata of each node & way traversed. My intuition says to use NoSql because the data is sparse and unstructured. In doing this we will reduce the number of API calls that we make and it will allow us the speed up computation as the number of iterations progress. Alternatively, I may consider moving all OSM data to MongoDB prior to running the algorithm if the time requirement is not achieved.

Mirror the proof of concept to capture the actual geospatial data. Experiment with parameters and measure the success of algorithm results according to section called Evaluation Metrics.