Machine Learning Engineer Nanodegree

Capstone Proposal

David Howard October 9, 2017

Proposal

Domain Background

SLAM (simultaneous location and mapping) is a well known technique for using sensor data to create a map of a geographic region and also keep track of the sensors location within the constructed map. Typically this is done with either a particle filter or a Kalman filter. One application of SLAM operates on an unknown region and builds up knowledge about the region via sensor measurements. Another application uses existing knowledge, such as maps, along with sensors, and is aimed at determining the sensor's location within the known region. (https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping (https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping))

In reading about SLAM, I found it interesting that I had previously been trained to be a human SLAM algorithm when I was a helicopter pilot in the U.S. Army. We trained to navigate high speed low level flight by using only a topographic map and our eyes observing the terrain elevation features (no GPS allowed). There is a name for this technique, Nap-Of-the-Earth flight. (https://en.wikipedia.org/wiki/Nap-of-the-earth#Helicopter_NOE_flying (https://en.wikipedia.org/wiki/Nap-of-the-earth#Helicopter_NOE_flying))

One source of potential data for use in a SLAM system is a digital elevation model, generically referred to as DEM. A great source of such data was created by the NASA Shuttle Radar Topography Mission (SRTM). In 2000, the Space Shuttle was used to create a digital elevation map of the entire world (minus some polar regions). The NASA SRTM data is publicly available for download, in various height resolutions. (https://www2.jpl.nasa.gov/srtm/mission.htm (https://www2.jpl.nasa.gov/srtm/mission.htm))

Problem Statement

This project will attempt to create a simple SLAM type implementation using a convolutional neural network to determine position within a known region. The train/test data used for learning will be SRTM elevation data treated as an image. Input for evaluation will be images composed of pieces of the same data with possible modifications such as distortion, alignment and reduced resolution. Output will be a predicted position of the input images. The metric will be accuracy of the predictions over a test data set.

I did not research whether or not this has already been attempted, but I assume it has been studied. I purposely did not look for any existing research like this keep my approach independent.

Datasets and Inputs

SRTM data is available online at (https://dds.cr.usgs.gov/srtm (https://dds.cr.usgs.gov/srtm)) . SRTM data is available in multiple resolutions. The data is segmented into 1 latitude/longitude degree squares. The highest publicly available resolution is 1 arc-second per elevation posting (Level 2, ~30 meters at the equator), which

results in a 3600x3600 matrix of elevations. The data files are actually 3601x3601 in order to fill overlaps if multiple cells are composed together. Other resolutions are 3 arc seconds (Level 1, ~90 meters) and (Level 0, 30 arc seconds).

A detailed description of the data is available at (https://dds.cr.usgs.gov/srtm/version2_1/Documentation/SRTM_Topo.pdf (https://dds.cr.usgs.gov/srtm/version2_1/Documentation/SRTM_Topo.pdf)).

Attached is an image created from a Level 2 cell converted to a ping and resized for email, for reference.

Solution Statement

The solution approach will be to use a convolutional neural network to identify the location of pieces of elevation data within a 1 degree SRTM data cell using CNN image recognition techniques. The solution will look something like the ML Nanodegree dog image project.

First, a one degree cell with a high variance in elevation (Mountains) will be selected from the available SRTM data. I expect this would help differentiate features in the data.

That input data will be preprocessed into an appropriate numeric format. The input will then be divided into individual rectangles of some smaller size that will be treated as is they were input images. The input images will be labeled with something representing the location within the 1 degree cell.

After training, predictions will be made using similar small squares of elevation data. Initially this test data will be identical to some of the training inputs. This is to test whether or not the model can predict anything at all. Since the test data will be a subset of the training data, if the model is constructed correctly it should have a high prediction accuracy.

Once that is working well enough, that model will be iterated again with more independent test data created by various distortions of the test data and the results evaluated.

Arriving at a solution, if it works at all, will involve experimentation with the organization, size and resolution of the training/test data. Ideally the training data will be a full 1-arc second, 3600x3600 cell, however that may result in training time to be too long for the scope of the project. If it is, then I can back off the the 3-arc second data which is 1200x1200.

If and when a solution is working, the final model will be trained and tested against a cell from the SRTM data that has low elevation variance, such as somewhere in Kansas (flat).

Benchmark Model

In this case, lacking an independent model to compare against, the benchmark will be the difference between the naive model described above, where training and test data is identical. That model will be iterated until it has a high accuracy. That will then be the basis for comparison of a new model using distorted training and test data.

Evaluation Metrics

The outputs of the model will be predictions of the probabilities of a test image location matching one of the actual locations. The outputs will be a set of probabilities, like the dog project. The 'absolute' accuracy will be the average of the highest predicted probability being correct over the set of test data. The evaluation will also

attempt to evaluate if the actual location matches any one of the higher of the list of predicted probabilities, it if isn't the highest. Although it is out of scope for this project, this information could possibly be used in a particle filter of some sort with input data being an image moved over the terrain, modeling a flight or drive.

Project Design

The models will be constructed in python using Keras with Tensorflow-gpu backend.

Benchmark Step

- input a 1 degree of elevation data
- preprocess it into a numpy array of floating point values [0..1.0] with rectangular shape.
- divide the array into a set of N rectangles of the input data of identical size
- using Keras, set up a CNN with inputs of the input rectangle size and outputs representing probabilities of location over the 1 degree area
- train the model using all the input data
- test against a subset of the input data (this will result in an overfit model, but that is the purpose of this step)
- iterate model, hyperparameters and configuration of input images until predictions have a high probability of success.

Evaluation Step

- input a 1 degree of elevation data
- preprocess it into a numpy array of floating point values [0..1.0] with rectangular shape.
- divide the array into a set of N rectangles of the input data of identical size
- using Keras, set up a CNN with inputs of the input rectangle size and outputs representing probabilities of location over the 1 degree area
- train the model using all the input data
- create a test data set using a subset of the input data with distortions (rotations, overlaps, sizes, reduced resolution)
- iterate model and hyperparameters and evaluate accuracy

Training will include input data with and without additional image preprocessing (ImageDataGenerator), and other variations on distortions of the test data set. For example, extract a rectangle from the dataset that overlaps two or more of the training rectangles.