Tracking in Images

CS-490 (Research & Development)

Sarthak Mittal

Indian Institute of Technology, Bombay

December 12, 2022

Overview

Overview

Description

We have an image of a road on a map taken from above (example as shown below). We want to approximate it using linear segments. The successive segments need to be chosen based on the state of the previously chosen segment.

Figure 1: A sample road map image

Assumptions

- For simplicity of the model, we consider **gray-scale** representation of the images
- The road runs from left to right and does not turn back at any point (analogous to a **one-to-one function**)
- The noise is independent and identically distributed **(I.I.D.)**
- The number of line segments we need to approximate using is *exactly* **k**
- The horizontal width of each segment is constant
- The road (and the approximation) should form a **convex polygon** upon connecting end point to start point
- The model makes decisions **sequentially** by choosing next point depending on parameters set by previous decision and maintaining constraints for the next

Sample Input

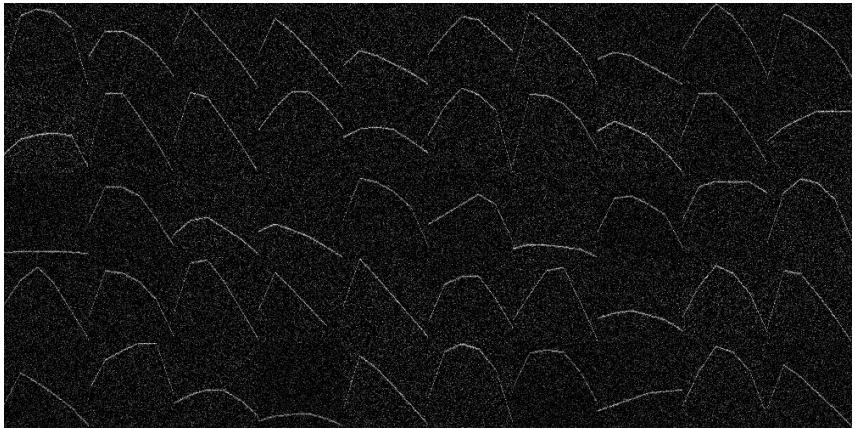

Figure 2: A randomly generated noisy input (50 samples of size 100×100)

Overview

Mathematical Setup

Consider the input as $X^{n \times w \times h}$, which is a set of n samples of road images of dimensions $w \times h$ (w inreasing to right, h increasing downwards) along with noise. We need to approximate the road in each sample using k linear segments.
Essentially, we need to find the set of points $P^{n \times k \times 2}$, where P contains the predicted points at ends of the k segments, such that horizontal difference is fixed.
Within a sample, say $X^{1 \times w \times h}$, based on the points predicted till some state, say $(x_i, y_i) \in P^1$, the region where the next point can be chosen will be restricted as:

$$m \leq \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \leq M$$

where M and m are defined as:

$$M = \frac{y_0 - y_i}{x_w - x_i},\ m = \frac{y_h - y_i}{x_w - x_i}$$

States & Parameters

We model the state of the system as having the following properties:

- The current position $S^{n\times 2} = \begin{bmatrix} x_{0i} & y_{0i} \\ \vdots & \vdots \\ x_{ni} & y_{ni} \end{bmatrix}$ where $0 \leq i \leq k$
- The loss accumulated so far by the path compared to sample
- The constraint parameters like M and m set by previous decision

The path till the current position is given as $P^{n\times i\times 2} = \begin{bmatrix} (x_{00}, y_{00}) & \dots & (x_{0i}, y_{0i}) \\ \vdots & \vdots & \vdots \\ (x_{n0}, y_{n0}) & \dots & (x_{ni}, y_{ni}) \end{bmatrix}$

Since the decision of the next state only depends on the current state (function of $S^{n\times 2}$, M, m and some other hyper-parameters), this can be modeled as a Markov Decision Process.

Forward & Loss Function

For the forward function of the model:

- The next state will be chosen based on the range of values (specifically range of y coordinates) allowed by the constraints
- The Gaussian distribution over the allowed points will decide the next pick
- We will choose the point which is sampled maximum number of times
- The parameters $\mu^{n\times k}$ and $\sigma^{n\times k}$ related to the choice distribution will be part of the model

At some state, we have the grid of the noisy sample whose diagonal is the estimated segment joining the current point to the next choice. The loss function can be calculated by computing some function that checks the points on the segment that lie within the region of the road. We can accomplish this using Linearly Constrained Neural Networks.

Linearly Constrained Neural Networks

We can combine a neural network with a linear programming problem. Since our constraint is in a linear format, we can assume we have 3 arrays of variables, m, M and y, all of size k. We can add a constrained optimization layer on top of the neural network, which would be a solver, for example Integer Linear Programming (ILP). In forward pass, we solve the current approximated problem (to find y, m and M that satisfy all constraints) and in the backward pass we can improve the approximated problem (based on loss accumulated and accordingly adding more constraints and eliminating lesser desired/noisy results).

References for LCNN:
https://sarthakmittal92.github.io/assistantships/cs490-2022/cs490-2022-ref.pdf.