Earth Mover's Distance based Similarity Search at Scale

Hey. This is some notes for an exam I have coming up. There's a real chance that my understanding here Is wrong. So take it with a grain of salt.
Also, they're written primarily for my own use. So they might not be 100% coherent.

The notes are talking about this paper:
https://cadmo.ethz.ch/education/lectures/FS18/SDBS/papers/p313-tang.pdf

Motivation

First of, why do we even want to use earth movers distance?

In some circumstances we want to compare different distributions/histograms/objects such as an image. Let's go with the case of an image, because that is easy to imagine (heh).
Let's say the image has three dimensions: red, green and blue pixels. Then image 1 might have 3 red, 2 green and 4 blue pixels. Image 2 might have 2 red, 5 green and 4 blue.

The question we now want to answer is, how similar are these two images to each other? One way of doing this, could be to plot them out as points in a euclidean space and then measure the distance between them. This is however not a robust way of doing it, as outlier values might cause the points to be far away - hence we can use the earth movers distance instead, which is a more robust measure.

The earth movers distance (EMD - so close to EDM, shame) is the minimum cost of transforming one object into the other. i.e. what would the min-cost be of transforming image 1 into image 2.

What can we use EMD for?

A very popular query to use with EMD is the k-nearest neighbours query, where k is the amount of neighbours we want to find. So if we have a database of images, and we got one of a puppy, we might want to find 3 other puppy images; then we carry out a 3-nearest neighbours search, by finding the 3 images with the lowest EMD when compared to our original image. Neat!

Ok, but how do we calculate EMD then?

There's various techniques for calculating the EMD between two objects. Both linear programming techniques and then there's min-cost flow techniques. It's the latter the paper talks about.
More concretely, they talk about the successive shortest path algorithm (SSP).

To understand SSP, we need to have a couple of definitions:

- Source nodes: these represent dimensions of the original histogram. For example the images would have three source nodes: red, green, blue.
- Target nodes: same as with source nodes, just for the histogram we're comparing to.
- Capacity: each node has a capacity. For the source nodes it's the amount of units it has to send to the target histogram in order to look like the target. For the target nodes it's the amount of units it have to receive from the source nodes, to look like the source.
- The cost matrix, C: this defines the cost of sending from one part of the bipartite graph to the other. Imagine this as telephone lines. Doing a national call will be cheaper than doing an international call. So if you can get away with doing your business just using national calls, that

would be preferable.

SSP works by:
1. constructing a complete bipartite graph between the two objects we want to compare (potential problem no. 1). Each of the source nodes have some capacity they want to send to the target nodes. The target nodes have some capacity they want to receive i.e. the difference between the two histograms.
   1. We then select a source node that has some remaining capacity.

Older

10th February 2024

Program@Stibo Gym