Unsupervised Detection of Metropolitan Areas

Anonymous Author(s)
Affiliation
Address
email

Abstract

City boundaries can be crisp or fuzzy depending on the effort that local governments put into digital mapping. The concept of a metropolitan area is even fuzzier than the concept of a city. This paper presents an usupervised algorithm to detect metropolitan areas from geographical data that is dense in urban areas and sparse in rural areas. As an example, we detect metropolitan areas for the UK using the density of crime. It is possible to tessellate Earth using metropolitan areas by associating to each point the metropolitan area closest to it. This can potentially aid geospatial algorithms by supplying a smart divide and conquer strategy.

1 Introduction

Geospatial algorithms are often designed with the human in mind. A few examples include (a) Resource allocation problems, such as determining the optimal locations for police stations, (b) Recommendation problems, such as ranking the most popular tourist destinations, and (c) Estimation algorithms, such as predicting real estate prices. Ideally, these problems would be solved per city, however, humans interact with their environment in ways that cut across city boundaries. In one extreme, a sprawling metropolitan area has no clear boundary where the "urban" ends and the "rural" starts. In another extreme, many cities can be close enough together so as to form a continuum of urban landscape that can stretch hundreds of miles. When one wishes to solve a problem that is local to a metropolitan area, it is not clear how to define the boundary of that metropolitan area.

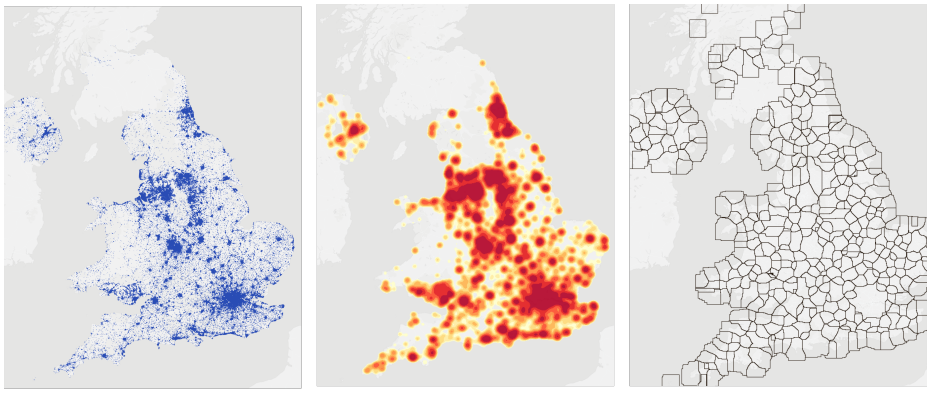
It is natural to wish to cut between metropolitan areas along lines which will least interfere with the problem we are trying to solve. For the police station allocation example, this might be interpreted as a cut between cities along a contour line that has minimal crime. For the recommendation example, this might be interpreted as a cut along along a contour line that is farthest from tourist attractions or from hotels. For the prediction example, this might be interpreted as a cut along a contour line where building density is minimal.

We present in this paper an algorithm to find metropolitan areas. The input is any geospatial variable that can be represented as a density per latitude and longitude. For the police station case, this can be the density of crime; for the recommendation case, this could be the density of tourist attractions or hotels; for the estimation case, this could be the density of buildings. The algorithm cuts between metropolitan areas along contour lines where the variable is least dense.

This algorithm is based on a technique from the field of image processing known as blob detection [ref]. Specifically, Lindeberg's blob detection algorithm [ref]. In this context, the density of, say, crime forms blobs when visualized as a heat map. The blobs naturally correspond to metropolitan areas. We use smoothing to introduce a minimal size for a blob to be considered a metropolitan area.

2 Metropolita area detection

In this section we describe our metropolitan area detection algorithm. The input is a list of latitudes and longitudes. We will explore in this section a specific example: a list of locations of crimes in the UK [ref].



(a) (b) (c)

Figure 1: Metropolitan area detection applied to crimes in the UK. (a) Locations of 2 million crime events reported in January–March 2018. (b) Smoothed heat map of crime, using Gaussian smoothing of radius $\sigma = 4$ km. (c) Metropolitan areas detected by Lindeberg's blob-detection algorithm. We plot lines at the boundary between pixels which do not share the same metropolitan area index.

We start by projecting the points into a local Cartesian coordinate system using the sinusoidal projection [ref] centered around a point $(\phi_0, \lambda_0) = (53°N, 2°W)$ which is roughly in middle of the UK. We do this so that we can use image processing techniques, such as Gaussian smoothing [ref], in the Euclidean plane. It is possible to do Gaussian smoothing on a sphere [ref] but when dealing with problems that are smaller than a large continent, a local Cartesian approximation works fine.

After projecting the points to a local Cartesian coordinate system, we pixelate the UK using a grid spacing of 1 km. This is measured at (ϕ_0, λ_0) and varies with latitude on a sphere, but is fixed in the Cartesian approximation. Within each pixel we count the occurrences of crime. We smooth this image using a Gaussian filter of radius σ. If two blobs are separated by a distance much smaller than this, they will be smoothed into the same blob.

Lindeberg's blob detection algorithm is run on this image to enumerate the blobs and mark the contour lines that delineate a blob from its neighbors. The process is to iterate on pixels in decreasing order of crime density. If a pixel has no higher-density neighbors, it is assigned a new "metropolitan area index." If the pixel has a neighbor with a higher density of crime, it is assigned the metropolitan area index of the neighboring pixel with highest density. Some care must be taken when there is a plateau of equal density: first expand to fill the plateau, then the entire plateau obtains a new index or copies the index of its highest-density neighboring pixel.

An example is shown in Figure [ref]. We downloaded UK crime data for the months of January–March 2018 from [ref], dropping crimes which did not have a latitude or longitude reported. There were 2 million crimes with valid location reported in this date range. As points, they are shown in [ref]. As a smoothed heat map they are shown in [ref]. Finally, the output of the blob detection step is shown in [ref].

3 Comparison across datasets and smoothing parameters

We wish to get a feeling for the stability of this algorithm, and the robustness of the term "metropolitan area" as derived from the algorithm discussed in this paper. To that end we compare the output of the algorithm across different datasets, and with different smoothing parameters. We will compare across three open datasets: (1) The UK crime dataset described in the previous section, (2) An index of place names in the UK [ref], maintained by the office for national statistics, and (3) A list of all highway nodes in Open Street Maps [ref] for the UK [ref].

The term "metropolitan area" also depends on the smoothing parameter σ. Increasing σ will decrease the number of metropolitan areas detected, and will increase the average area of each metropolitan area. We will compare the three values σ: 5km, 10km, and 20km.

There is a third parameter discussed in Section [ref], the grid size. Here, smaller is better, with the only tradeoff being that smaller grid sizes increase the computational time needed for the algorithm to run. For the visualizations created in this paper, a grid size of 1 km provided sufficient detail, and the algorithm ran in a

Figure 2: Comparison of our metropolitan area detection algorithm across different datasets and smoothing parameters. We show the boundaries between metropolitan areas for each of three datasets discussed in the text, and for each of $\sigma = 5, 10, 20$ km. We also overlay the figures across datasets for each value of σ to show the stability of metropolitan areas with respect to the dataset.

few minutes for each of the three datasets on an Intel i7 laptop. The algorithm was implemented in python and we made no attempt to optimize it.

The result of the comparison is shown in Figure [ref]. The three datasets are labeled "crime", "places", and "roads", respectively. We show the boundaries between metropolitan areas for each dataset and for each value of σ. We also overlay the plots across datasets for each value of σ to show that the metropolitan areas are stable across datasets. The crime dataset had 2 million rows, the places dataset had 87 thousand rows, and the roads dataset had 28 million rows.