Geospatial Analytics 24-25

Notes

University of Pisa

M.Sc. in Data Science and Business Informatics

Contents

Chapter 1

Introduction

A geographic information system (GIS) is a computer system used to capture, store, query, analyze, and display geospatial data. **Geospatial data** describes both the location and the characteristics of spatial features: for example, if we want to describe a road, we may refer to its location and its features (length, name, speed limit, etc.). Other than single entities, geospatial data can also describe trajectories, specifying the sequence of locations constituting it.
The following chapter will give an overview of the basic concepts used in GISs and spatial data analysis.

1.1 Geographic Coordinate Systems

When using a GIS, any map layers used together must align spatially; to make sure this is true, we need to use some common spatial reference system for all maps. GIS users normally work with locations expressed on a plane using a coordinate system expressed in x- and y-coordinates, while the actual, real-life locations represented by them are on Earth's surface (which is ellipsoidal). A **map projection** is used to convert the Earth's surface to a plane.

The system used to locate points on Earth is called **geographic coordinate system**. This system is defined by two coordinates: **longitude** and **latitude**. They are angular measures which measure the angle at which the point can be found with respect to the **prime meridian** and the **equator**; longitute represents the angle east or west from the prime meridian, while latitude represents the angle north or south of the equatorial plane.

Meridians are lines of equal longitude. The prime meridian passes through Greenwich, England, and corresponds to 0°. **Parallels** are lines of equal latitude. The equator is the line corresponding to 0° latitude.

In a **plane coordinates** system, longitute and latitude correspond to x and y coordinates respectively. Logitude takes positive values in the easter hemisphere, and negative values in the western hemisphere; latitude takes positive values north of the equator, and negative values south of the equator.

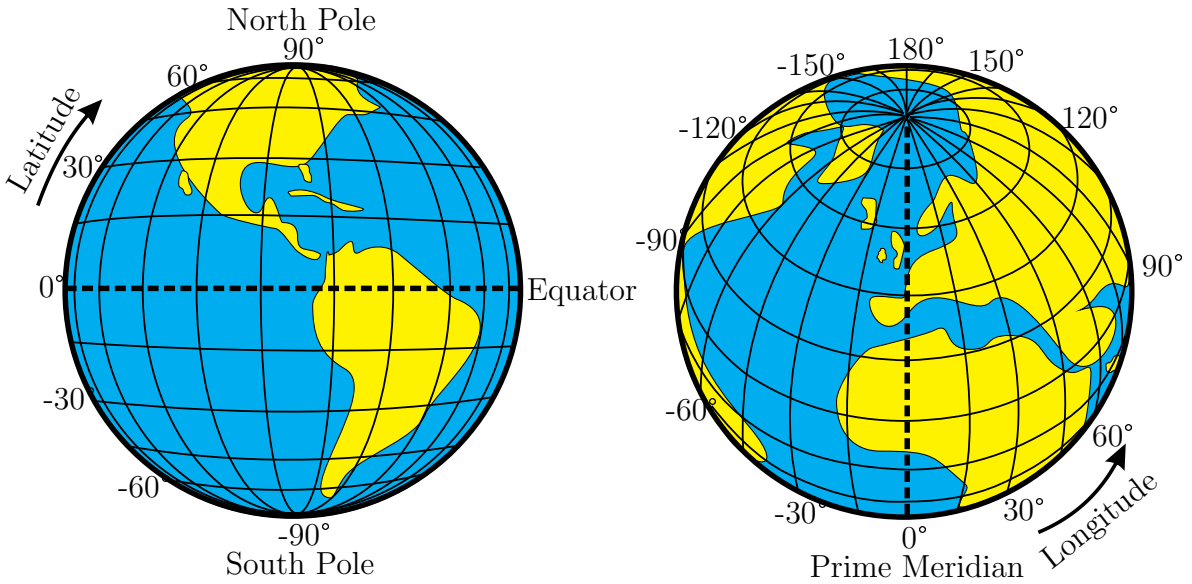

Figure 1.1: Latitude and longitude.

Longitute and latitude values may be expressed in different ways:

- **Decimal degrees** (**DD**): represented by a single decimal value;
- **Degrees-minutes-seconds** (**DMS**): represented by a set of three values, corresponding to degrees, minutes, and seconds. 1 degree corresponds to 60 minutes, and 1 minute corresponds to 60 seconds;
- **Radians** (**rad**): similar to DD, but expressed in radians instead of decimal values: 1 degree is equal to 0.01745 rad.

As mentioned before, planet Earth can be approximated as an **ellipsoid**: this shape is obtained by rotating an ellipse by its shortest axis. Indeed, Earth is wider along the equator (its major axis) than it is between the poles (its minor axis). Another parameter that describes an ellipsoid is the **flattening** (f), calculated as $f = \frac{maj-min}{maj}$, and it describes the difference between the two axes.

A **datum** is a mathematical model of the Earth which is used as the reference to calculate the geographic coordinates of a point (or even the elevation, if we consider vertical datums). A datum is defined as: the pair *longitude, latitude* of coordinates of

an initial point wich will be the origin, an ellipsoid, and the separation of the ellipsiod and the Earth at the origin.

Distances on the Earth's surface are not straight lines; they are instead represented by **geodesics**: through any two points (not antipodal), there is exactly one "great circle" that connects them. The two points separate the great circle in two parts: the shorter of the two is their geodesic distance. Since the Earth is nearly spherical, geodesic distances are correct with an error up to 0.5%. The geodesic distance between points A and B is calculated using the **spherical law of cosines**:

$$\cos(d) = \sin(lat_A)\sin(lat_B) + \cos(lat_B) + cos(lon_A - lon_B),$$

where d is the angular distance between the two points. To convert d to a linear distance measure, it can be multiplied by 111.32 km, which is the length of 1 degree at the equator. This formula is the basis from which the **haversine formula** is derived.

1.2 Trajectories, Tessellations, Flows

Let u be an individual. A **trajectory** $T_u = \langle p_1, p_2, \ldots, p_{nu} \rangle$ is a time-ordered sequence composed by the spatio-temporal points visited by u. A spatio-temporal **point** is a pair $p = (t, l)$, where t is the time, and $l = (x, y)$ is the point visited at that time.

Given an area A, a **tessellation** is a set of geographical polygons with the following properties:

- It contains a finite number of polygons called **tiles**:

$$\mathbb{G} = \{g_i : i = 1, \ldots, n\}$$

- The tiles are non overlapping:

$$g_i \cap g_j = \emptyset\,, \forall i \neq j$$

- The union of all tiles completely covers the tessellation:

$$\bigcup_{i=1}^{n} g_i = A$$

A tessellation can be **regular** or **irregular** depending on the shape of its tiles. Regular tessellation may use equilateral triangles, squares, hexagons; irregular tessellation may use buildings, census cells, administrative units. A **spatial join** is used to associate a point with the tile it belongs to. Since the tiles are non overlapping and cover the

entire area, each point belongs to one and only one tile. **Voronoi tesselations** are a particular type of tessellation that partition the plane into regions (called **cells**), each closer to a secific point (called **seed**) out of a set. Each of these cells is defined as the set of points that are closest to the seed of the cell itself than any other seed in that area.

Given a tessellation, the **flow**

$$y(g_i.g_j)$$

represents the number of people/objects moving between g_i and g_j. A trajectory refers to a single entity, while a flow refers to the total amount of entities moving between two points. Flows can be derived from a set of trajectories, but the inverse is not true.

1.3 Raster and Vector Data Models

The raster and vector data models are two ways to represent geographic information in GISs. In both cases, data can be stored in several **thematic layers**, each of which contains a set of objects of the same nature. For example, a layer may contain information about buildings, another about streets, another about rivers, and so on.

The **raster data** model divides the space into a regular grid of square cells with a given size (which defines the **resolution**). This format is often used for images, where each element corresponds to a pixel. Depending on how much information is assigned to a cell, data can be **single-band** (one attribute per cell), or **multi-band** (several attributes per cell). Raster data is typically sourced from satellites.

The **vector data** model uses discrete objects (points, lines, polygons) to represent spatial features. Each object can have its own properties and relationship with the others. A **point** is a zero-dimensional object with a *location* property (expressed as x,y coordinates). A **line** in a one-dimensional object with two properties: *location* and *lenght*. It be either straight or curved. A **polygon** is a two-dimensional object with three properties: *location*, *area*, and *perimeter*. These objects are expressed differently depending on the data format used by the software/platform.

Objects in a layer are sometimes also called *spatial features*; for this reasin, the variables associated to them are called *attributes* (and not features). To represent geometric objects in a GIS, we can use one of the following models:

- **Geo-relational data model**, where objects and attributes are stored separately, and associating each object to the corresponding attributes requires a join operation (at the advantage of possibly saving space if a certain attribute(s) is (are) only possessed by few objects);

- **Object-relational data model**, where objects and attributes are stored together in a single table, making retrieval much faster (but possibly increasing the amount of space needed to store everything).

In principle, vectors can model everything; raster data is a discretized view of the same information. Raster data is better suited for "dense" data; it can be more efficient in those cases where a raster representation may need a very high number of objects, but precision is not a concern. Vector data can also be converted to raster data, and vice versa. **Rasterization** is the process of transforming vector data into raster data, and produces a discrete approximation. **Vectorization** is the inverse process: it may be difficult at times, and many algorithms and methods have been developed to perform it.

At times, vectors and raster information can be used together in multi-layer data. Some information is better modeled with one format than the other: for example, street networks or locations of interest are often encoded as vector layers, while things like land usage are encoded as raster layers.

1.4 Spatial Operations

The most important spatial operations are:

- **Intersection**: returns all the points in common with the operands.
- **Union**: returns the union of the two operands. In some tools, They are kept as separate objects, meaning that the result is always a multipolygon. As an alternate operation, the same tools offer the **dissolve** operation, which instead merges the two objects into a single one.
- **Difference**: returns all the points in the first operand which are not in the second.
- **Buffering**: creates a buffer, i.e., an expanded area, around the object. The result is equivalent to replacing each point in the geometry with a circle with a given radius.
- **Spatial join**: like in relational databases, joins merge the information of two objects. THe join can be **inner** (the output contains only pairs in common with both objects), or **outer/left/right** (the output also contains non matching objects with the *NULL* value in place of the missing attributes).

1.5 Spatial Patterns and Spatial Correlation

Point Pattern Analysis (**PAA**) is the study of point patterns, i.e., the spatial distribution of points in an area. Spatial distributions are typically categorized into three types:

- **Uniform (discrete)**: points are evenly distributed in the area;
- **Random**: points are distributed according to a random process;
- **Clustered**: points appear to be grouped (clustered) in some areas.

A basic form of point pattern analysis consists in determining summary statistics such as mean center, standard distance, and standard deviational ellipse.
Mean center is the average of the x and y coordinate values:

$$\bar{s} = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n} \right)$$

Standard distance measures the variance between the average distance of the features to the mean center:

$$d = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu_x)^2 + (y_i - \mu_y)^2}{n}}$$

Similar to standand distance, **standard deviational ellipse** measures the standard distances for each axis:

$$d_x = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu_x)^2}{n}}$$

$$d_y = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \mu_y)^2}{n}}$$

Average Nearest Neighbor (**ANN**) is an algorithm that can be used to study patterns. For each point, its nearest neighbor is found as the point with the smallest distance to it. The average of all points' nearest neighbor distance is calculated as d_{obs}, and normalized with regards to the expected average if the pattern were random (d_{exp}), obtaining a ratio:

$$R = \frac{d_{obs}}{d_{exp}}$$

If $R = 1$, the pattern is random. If $R < 1$, the pattern is clustered, because the distances are smaller than expected; if $R > 1$, the pattern is uniform (or at least more dispersed than random).

Ripley's K-function is another popular method for analyzing point patterns. Usually, its normalized version, called **L function**, is used. Given n points in an area of size A, and a distance d, the L function is calculated as follows:

1. Compute all $n * (n-1)$ distances between each pair of points;
2. Compute ϕ, the fraction of distances that are $\leq d$;
3. Compute

$$L(d) = \sqrt{\frac{A}{\pi}\phi}.$$

$L(d) = d$ for random distributions. If $L(d)$ is higher, the data is more clustered; if it is lower, the data is more dispersed. Different values of d can be explored to understand patterns at different spatial granularities.

Another important aspect is **density based analysis**. Density measurements can be either global or local. **Global density** is simply calculated as the ratio of observed points and the study region's area:

$$\hat{\lambda} = \frac{n}{A}$$

Density can also be measured at different locations of the study region. **Local density** is computed over a single tessellation cell; the chosen resolution will affect the resulting density calculation. **Kernel density** is another method of calculating density per-cell which considers also the points found in its neighborhood. Usually, given a cell c, the 8 adjacent cells are considered as the neighborhood N_c, and so the kernel density becomes:

$$\textit{Kernel density } (c) = \hat{\lambda}(c \cup N_c)$$

A variant is **weighted kernel density**, which assigns to each point a weight inversely proportional to the distance from the cell's center. Different weight functions can be used; a common one is the Gaussian function.

Autocorrelation is the correlation of the values of a same variable measured at different points in time (**temporal autocorrelation**)/space (**spatial autocorrelation**). An example of spatial autocorralation may be checking how much the temperature values in the points of a layer are influenced by the neighboring values. According to **Tobler's first law of geography**, "everything is related to everything else, but near things are more related than distant things": this is the fundamental assumption in spatial analysis.

Popular measures of spatial autocorrelation are:

- **Moran's I**, which calculates the autocorrelation between values of each point against all other points in its neighborhood:

$$I = \frac{\sum_{i=1}^{n} \sum_{j:x_j \in N_{x_i}} w_{ij}(x_i - \mu_x)(x_j - \mu_x)}{s^2 \sum_{i=1}^{n} \sum_{j:x_j \in N_{x_i}} w_{ij}}$$

where s^2 is the variance of the x values, and w_{ij} is a weight, typically defined as the inverse of the distance between the two points. Positive values mean positive correlation, negative values mean negative correlation;

- **Geary's C**:

$$C = \frac{n-1 \sum_{i=1}^{n} \sum_{j:x_j \in N_{x_i}} w_{ij}(x_i - x_j)^2}{2(\sum_{i=1}^{n} \sum_{j:x_j \in N_{x_i}} w_{ij}) * \sum_{i=1}^{n}(x_i - \mu_x)^2}$$

  The higher it is, the more different are nearby values (less correlation), the lower it is the closer they are (more correlation).

Both measures can also be interpreted as the average of local I/C values calculated across neighborhoods. These local values can be studied individually as well.

1.6 Spatial Interpolation

Spatial interpolation refers to the process of using points with known values (called **control points**) to estimate values at others. For example, we could estimate the temperature at a point with no recorded data by approximating it from known temperatures at nearby points. Ideally, control points should be well distributed across the study area, although this situation is rare in real-world applications since a study area oftentimes also contains data-poor areas.

Interpolation methods can be divided in two groups: **deterministic** and **stochastic**. The first group assumes that the known values are exact, with no assessment of errors for predicted values. The second group considers the presence of some random error in the known data and offers some assessment of prediction error with an estimated variance.

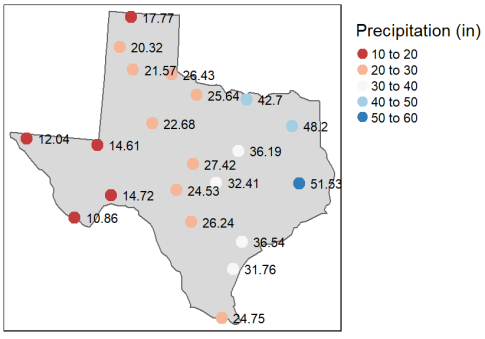

Figure 1.2: An example of samples corresponding to yearly precipitation recorded in different sites in Texas.

1.6.1 Deterministic Methods

Proximity Interpolation

Proximity interpolation (also known as **Thiessen interpolation**) is one of the simplest and oldest interpolation methods. The goal is to assign to all unsampled locations the value of the closest sampled location, producing a Voronoi tessellation over the study area. All the points within the same cell have the same value. A problem of this approach is that surface values change abruptly across the perimeter of adjacent cells, which is not realistic.

Inverse Distance Weighted Interpolation

Inverse Distance Weighted (IDW) interpolation calculates an average value using nearby weighted locations. The weight of each sample location is inversely proportional to the distance; the value at location j is given by:

$$\hat{Z}_j = \frac{\sum_i Z_i / d_{ij}^n}{\sum_i 1/d_{ij}^n}$$

Here, d_{ij} is the distance between points i and j, and n is an hyperparameter that controls the irrelevance of a point as the distance increases/decreases: the larger n is, the less far away samples influence the interpolated value. For $n \to \infty$, the result is equivalent to proximity interpolation.
Values returned by this method are always within the range of the known values: $[Z_{min}, Z_{max}]$.

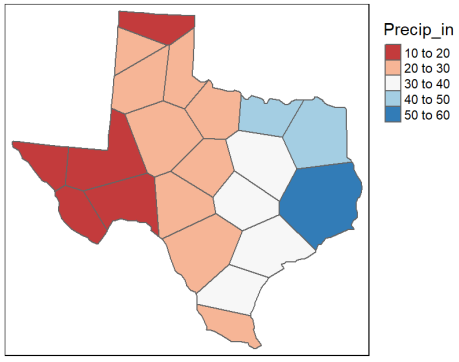

Figure 1.3: Interpolated values obtained by proximity interpolation.

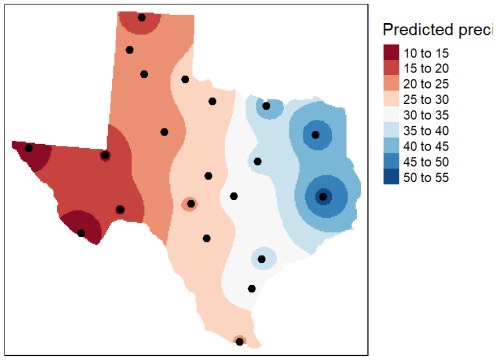

Figure 1.4: Interpolated values obtained by IDW interpolation.

1.6.2 Stochastic Methods

Trend Surface Interpolation

Trend surface analysis approximates points with known values using a polynomial equation. The same equation can then be used to predict values at other points. Depending on the order of the polynomial, the approximation can be more or less complex:

- A 0^{th} order surface is described by $Z = z$, where c is the average value of all samples;
- A 1^{st} order surface is described by $Z = aX + bY + z$, where X, Y are the coordinate pairs and c is a constant;
- A 2^{nd} order surface is described by $Z = aX^2 + bY^2 + cXY + dX + eY + z$;

and so on. Changing the order allows the model to better capture the complexity of the data, but using a value that is too high may result in overfitting the data, meaning that the model is too dependant on the known information and does not provide a useful prediction.

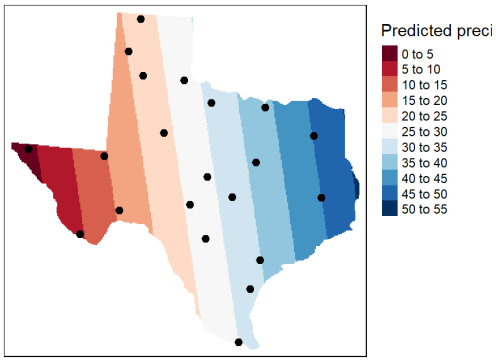

Figure 1.5: Interpolated values obtained by a 1^{st} order trend surface. The model is too rigid.

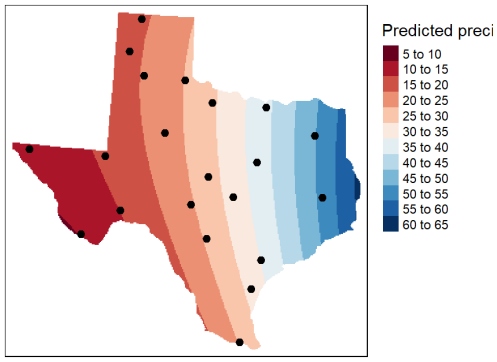

Figure 1.6: Interpolated values obtained by a 2^{nd} order trend surface. The model is better than before, but still not a good fit.

Kriging

Kriging differs from other methods in that it can assess the quality of prediction with estimated prediction errors. It assumes that the spatial variation of an attribute is neither random nor deterministic; instead, it is a combination of some spatially correlated component, a "drift" (assumed for now to be null), and a random error term.

The first step is **de-trending the data**, so that mean and variance of the data are constant across the whole study area. This can be done by using any trend model and subtracting the predictions from the data. From this point on, the method will focus on residuals, i.e., the remaining variability in te data that is not explained by the global trend. The second step is **constructing a (semi)variogram**. For each pair of points i and j, their semivariance is calculated as:

$$\gamma = \frac{(Z_i - Z_j)^2}{2}$$

The variogram is obtained as the plot of all the semivariances, using the x axis to represent distances, and the y axis to represent the semivariances. Usually, it is simplified by binning over the distance values and averaging the semivariances within each bin, obtaining a **sample experimental variogram**. The third step is to fit an **experimental variogram model** to find the parameters that best describe the spatial variance; there are many model variants, with the main ones being the Gaussian, linear, and spherical ones. Finally, **interpolation** can be performed. The general equation for estimating the residual at a point j is:

$$z_j = \sum_{i=1}^{s} z_i W_i$$

where s is the number of sample points used in the estimation, and W_i is the weight assigned to point i. These weights are derived by solving a set of simultaneous equations.

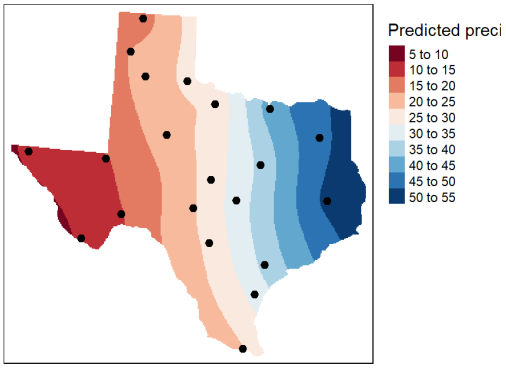

Figure 1.7: Interpolated values obtained by kriging.

1.6.3 Spatial Regression

Regression has the goal of modeling the relationship between a target variable and a (set of) independent variable(s). The difference with interpolation is that in the latter,

only the target variable data and the spatial coordinates are used, while in regression there is information about other non-spatial attributes to make predictions.

Spatial regression extends the typical regression method to include information about a point's neighborhood:

$$(P_i) = \underbrace{\alpha + \beta X_i + \epsilon_i}_{\substack{\text{Standard regression} \\ \text{model}}} + \underbrace{\delta \sum_j w_{ij} X_j}_{\substack{\text{Summary information} \\ \text{about neighbors}}}$$

1.7 Co-location Pattern Mining

Co-location pattern mining is the process of finding subsets of spatial features whose instances are frequently located together in space. It is similar to traditional frequent itemset mining, although defining "transactions" is not as immediate, since there is no explicit way to isolate instances into groups.

Given:

- A set of features $F = \{f_1, \ldots, f_m\}$;
- A set of instances $O = \{o_1, \ldots, o_n\}$;
- A neighbor relation R between pairs of instances, s.t. $R(o_1, o_2) \iff d(o_1, o_2) < \textit{thresh.}$;

a **co-location pattern** $CL = \{f_1, \ldots, f_k\} \subseteq F$ is a subset of features, and an **instance of a pattern** $I = \{o_1, \ldots, o_k\} \subseteq O$ is a subset of objects s.t.:

- For each $f \in F$, there is exactly one object $o \in O$ of type f, anc vice versa;
- I forms a clique w.r.t. R (i.e., all pairs of objects are connected to each other).

A co-location pattern that contains k spatial features is called size-k co-location pattern.

To measure the prevalence of a given pattern, we use two measures; the **participation ratio** (of a feature within a pattern):

$$PR(CL, f_i) = \frac{\#|\pi_{f_i}(\textit{instances(CL)})|}{\#|\textit{instances(CL)}|}$$

and the **participation index** (of the entire pattern):

$$PI(CL) = \min_{i=1}^{k} PR(CL, f_i)$$

A co-location pattern is called **prevalent** if its partecipation index is greater or equal than a given threshold *minprev*. The participation index is anti-monotone, which means that the participation index of a co-location pattern is always lower or equal to that of its sub-patterns.

1.8 Trend Detection

Spatial trend analysis is the process of finding patterns of change of non-spatial attributes in the neighborhood of some object in the study area. It is analogous to trend analysis in time series, but applied to spatial data, where the linear direction of time is replaced by the many possible paths that can be formed in space. The most notable paths of interest usually start from a common location (e.g., a city center), have meaningful shape, and show a statistically significant trend.

Let g be the neighborhood graph that represents all the neighborhood relationships among objects in the study area. Let o be one such object, which is considered the starting point of paths. Let a be a subset of all non-spatial attributes, in which paths must be found in. Let t be a type of function (e.g., linear, exponential) used for regression. Then, the goal of spatial trend detection is to discover the set of all neighborhood paths in g, starting from o, having a trend of type t in attributes a, with a correlation greater or equal to a given threshold *minconf*.

To isolate these relevant paths, special filters are used to select subsets of all paths. There are many ways to define such filters; some examples are reported in Fig 1.8.

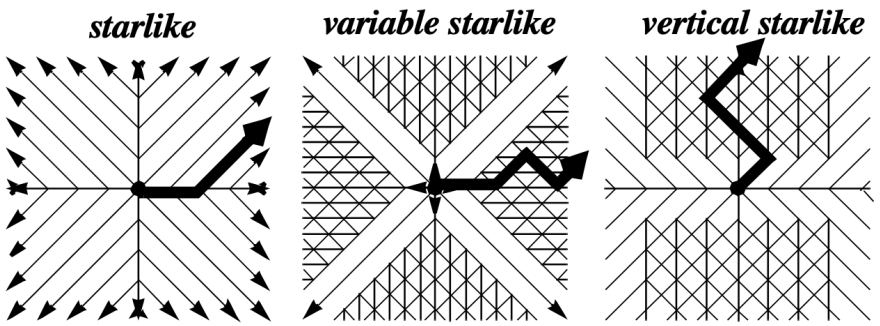

Figure 1.8: Examples of possible filters. The *stralike* filter allows paths that deviate in a specific diagonal direction; the *variable starlike* filter allows paths that can deviate diagonally multiple times; the *vertical starlike* filter restricts horizontal deviations in favor of vertical ones.

Chapter 2

Spatial and Mobility Data

The study of human mobility data has a wide range of applications, such as urban planning, epidemic control, transportation management, immigration monitoring. The last decade has seen a rapid increase of digital traces that represent human movements. Examples of this data are those generated by GPS devices embedded in phones or cars, mobile phone records collected by telco companies, and geotagged social media posts. This chapter will describe the main characteristics of such data.

2.1 GPS Data

Global Navigation Satellite Systems (**GNSS**) use satellites to provide geo-spatial positioning of objects on the Earth's surface in terms of longitude, latitude, and altitude. There are currently multiple systems managed by different countries of the world. The most famous GNSS is the US **Global Positioning System** (**GPS**), and receivers are now embedded in many commonly used tools such as smartphones, cars, and wearable devices.

GPS can be thought of a system of three main components: a space segment, a control segment, and a user segment. The **space segment** consist of a constellation of 24+ satellites orbiting around the planet, circling it twice a day in six equally spaced orbits. Each orbit has four satellites, and any user will always see at least four satellites from any point on Earth. The **control segment** consist of all the ground facilities used to track/command the satellites and monitor their transmissions. The **user segment** is represented by the GPS receivers in the user devices. On mobile phones, GPS receivers are activated by software that requires the user's position. On cars, receivers automatically turn on as the car starts. Signals are sent to a server every few seconds up to every few minutes; their precision can range from a few centimeters to meters, depending on the quality of the device.

To identify a point on Earth, a total of four satellites are used (three to identify the point, a fourth one for redundancy): each satellite broadcasts a radio signal at the speed of light that contains information about its location, status, and time of the on-board atomic clock. When the signal hits the device receiver, the time of arrival is compared to that of the send to calculate the distance d from the satellite.

Using this distance, we can imagine a sphere of radius d centered around the satellite. The intersection of all the spheres centered around the involved satellites is the exact location of the device.
The GPS traces format is as follows:

$$(u, lat, lng, alt, t)$$

where:

- u is the user/receiver ID;
- lat is the latitude;
- lng is the longitude;
- alt is the altitude;
- t is the timestamp.

Pros Ubiquitous, produces high resolution, dense traces.

Cons Rarely publicly available, no semantic information (e.g., what the user is doing, if he/she is in a building or outside), prone to errors when signal is noisy or absent, and small samples.

2.2 Mobile Phone Records

Mobile phone coverage reaches almost 100% of the population of most countries. Telco companies collect the activity of phone users for billing purposes, storing information about where users are located, when they communicate with others, and who they communicate with. This type of information can be split in the following three groups.

Call Detail Records CDR are generated each time a user makes/receives a call or an SMS. A CDR is a tuple

$$(n_o, n_i, t, A_o, A_i, d)$$

where:

- n_o is the caller's ID (can be the phone number, but it should be anonymized for privacy reasons);
- n_i is the receiver's ID (same as above);
- t is the timestamo if the call;
- A_o is the ID of the antenna the caller is connected to;
- A_i is the ID of the antenna the receiver is connected to;
- d is the duration of the call.

eXtended Detail Records XDR are generated when a device uploads or downloads data from the Internet. These connections can be both human-triggered and device-triggered (e.g., software updates, background synchronization). An XDR is a tuple

$$(n, t, A, k)$$

where:

- n is the user's ID;
- t is the timestamp of the connection;
- A is the ID of the antenna the user is connected to;
- k is the amount of KB of data transferred.

Control Plane Records CPR are used to check the status of the network, and are generated each time a device connects to a new antenna. These records are exclusively network-triggered. A CPR is a tuple

$$(n, t, A, e_1, e_2, \ldots, e_n)$$

where

- n is the user's ID;
- t is the timestamp of the connection;
- A is the ID of the antenna the user is connected to;
- e_i are the "events" that happen in the network.

Pros Ubiquitous, huge sample size, rich and multidimensional (give information about space, time, and social connections among users, since we can know who calls/messages who).

Cons Not publicly available, much more sparse and low resolution than GPS, suffers from ping-pong effect (since areas of influence of antennas are not perfectly regular and may overlap, a device may rapidly flip between them).

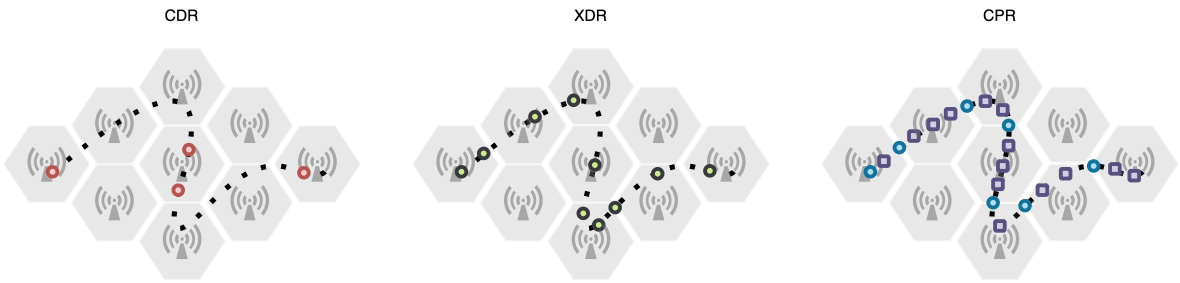

Figure 2.1: Comparison of densities of different data types.

2.3 Location-based Social Networks

Certain social media platforms allow users to geotag their posts, i.e., assign geographic and temporal information. This information can be represented by absolute data (latitude, longitude), relative data (distance from some point of interest), or symbolic (home, office, stadium, shopping mall). This geospatial information can then be easily coupled with the social network structures of connected users, allowing us to consider the social dimension in our analysis. A LBSN record is a tuple

$$(u, t, lid, lpos, post)$$

where:

- u is the user's ID;
- t is the timestamp of the post;
- lid is the location ID;
- $lpos$ is the location position;
- $post$ is the content of the text post.

Pros Ubiquitous, can provide precise information with little error, carries semantic information.

Cons Is very sparse (even more than mobile phone records), suffers from self-selection bias (users that frequently produce such content tend to belong to a specific demographic).

2.4 Road Networks and Points of Interest

Road networks describe streets and their intersections. They are usually represented by multigraphs, where a node is an intersection and an edge is a road. Nodes and edges

are geometries, encoded as Point, Lines, or Polygons. A road network is described by a two sets of records: the first contains all the nodes, the second all the edges. A node record is a tuple

$$(id, lat, lng, streets, geom)$$

where:

- id is the node ID;
- lat, lng are the geographic coordinates;
- $streets$ number of streets crossing in that intersection;
- $geom$ is the geometric shape of the node (usually a Point).

An edge record is a tuple

$$(id, u, v, f_1, \ldots, f_k, geom)$$

where:

- id is the edge ID;
- u is the origin node;
- v is the destination node;
- $f_i, \ldots, f_k$ are the road characteristics (e.g., if the road is one-way only, its length, number of lanes, name, speed limit);
- $geom$ is the geometric shape of the edge (usually a Line).

Road networks help associate GSP traces to actual streets, and can find routes between two places. These structuresa are also aften used in simulations for traffic management and urban planning.

Points of interest describe certain geospatial entities in a city, such as grocery stores, transit stops, restaurants, etc. They carry semantic information about activities in cities and associated human behaviour. A POI record is a tuple

$$(id, f_1, \ldots, f_n, geom)$$

where:

- id is the POI ID;
- $f_i, \ldots, f_n$ are the POI characteristics (e.g., name, type of building, road nodes involved);
- $geom$ is the geometric shape of the POI (usually a Point or a Polygon).

Chapter 3

Data Preprocessing

Data that represents trajectories of moving entities is often affected by errors. Their presence can greatly influence the result of analysis, so it is important to identify and remove them. Additionally, datasets tend to contain huge amounts of objects, meaning that the time and space complexity that may bee too high. Finally, individual trajectories that are too long and complex may cause the failure of data mining algorithms due to the inability of extracting details from them. Summarizing the information carried by trajectories via a preprocessing phase is a necessary step to make sure subsequent analysis can be carried out efficiently and effectively.

Preprocessing techniques can be divided into two families: context-based filtering, and movement-based filtering. **Context-based filtering** relies on contextual geographical information provided by the environment, such as road networks, to remove points that are deemed to be errors. These methods may not always be effective, since they assume that trajectories must always be on a road (or adapt in some way to the available data); this may not always be true, for example for vehicles moving across the sea or open fields, or for pedestrians. **Movement-based filtering** uses only the geometry and dynamics of the movement described by the trajectory to find points that do not match the expected behaviour of the moving entity.

3.1 Movement-based Filtering

3.1.1 Speed-based Filtering

This filtering is based on the idea that tha speed of the moving entity must always be within a certain range of allowed values, without any abrupt change. Initially, the first point of the trajectory is found and set as valid; then, each subsequent point is scanned in order according to the timestamp, and the speed v between the current and

the previous point is calculated:

- If v is larger than a given threshold, the point is marked as invalid and removed from the trajectory;
- If v is within the threshold, the point is marked as valid and kept.

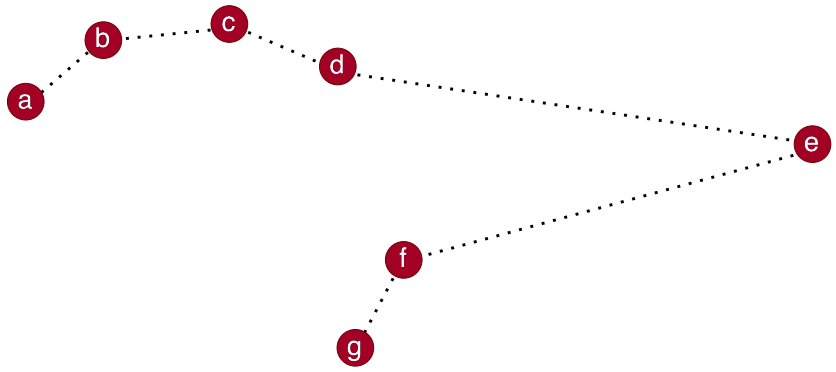

Figure 3.1: Example of speed-based filtering: point e is removed from the trajectory as it would imply a speed higher than the threshold.

3.2 Context-based Filtering

3.2.1 Point Map Matching

Map matching is the procedure of determining which road on a map a vehicle is using, given a set of GPS points. This approach both filters out noise and outliers, and adds new information to the dataset by associating each point with an identifier of the road it is matched to. Once points have been mapped to a specific road, any point farther than a certain threshold can be removed as noise.

The main issues of this approaches are:

- Projecting individual points requires a comparison for each point and each road segment, which can be computationally expensive;
- In some cases, the same point could be potentially assigned to multiple road segments, but limiting ourselves to consider the closest one may not always be the best choice;
- Matching points individually can lead to inconsistencies, especially when the road network is very dense.

3.2.2 Route Reconstruction and Trajectory Map Matching

Route Reconstruction Route reconstruction tries to infer the path taken by a moving entity by connecting the available points and possibly creating new ones; this technique is useful in cases where data is generally sparse or has large missing gaps. Usually, we assume the entity always takes the shortest (or best) path between two consecutive points; if an area allows free movement, the entity is assumed to move in a straight line. If there are limitations, such as the presence of buildings, the path must adapt to them. Since we often have attributes describing the characteristics of roads, the optimal path can be found by trying to maximize or minimize certain features: the total path duration in time, fuel consumption, CO_2 emissions, etc.

Common algorithms used for route reconstruction are **Dijkstra's minimum cost path algorithm**, which is efficient and can handle cycles, although it requires non-negative costs, and **Bellman-Ford's algorithm**, which is less efficient and can't handle cycles, but allows negative costs.

Trajectory map matching This is a more advanced version of point map matching that takes as input a whole trajectory (wheter it comes from raw data or from a reconstruction step), and adapts it to a road network considering spatial, temporal, and contextual constraints for the whole path. Common approaches are shortest-path based, and probability-based.

Shortest-path based Map Matching starts by matching only the first and last point of the path to the road segments, as in point map matching, and then finding the optimal path that connects them. A threshold is set as the maximum distance allowed between a segment and a GPS point for the two to be considered matching; using this threshold, **cutting points** are found. A cutting point can be of two types: the first type is the point that is the farthest from the current optimal path, and the second type are the points whose distance fall within the threshold. If a cutting point of the first type is found, the path is split in two, and a new optimal path that passes by that point is found. These steps run recursively on each half until all the points are reasonably close to the optimal path.

Probability-based Map Matching considers the probability that an entity may move from one road segment to another. State-of-the-art implementations use Hidden Markov Models to model the trajectory as a sequence of state transitions.

3.3 Trajectory Simplification

Many algorithms on trajectories tend to be very expensive, and their complexity is tied to the number of points in the dataset. Trajectory simplification is used to reduce the number of points without affecting the quality of results. Typical cases in which simplification can be used are for straight line movement (the entity keeps moving in a straight line, so the whole trajectory/subtrajectory can be represented as a single segment), or negligible movement (the entity stays still for some time, accumulating GPS traces in the same spot). Some standard methods for polygonal curve simplification include Ramer-Douglas-Peucker, and Driemel-HarPeled-Wenk.

Ramer-Douglas-Peucker is one of the most successful simplification algorithms, used in many fields other than GIS, such as computer vision and pattern recognition. A pseudocode of the algorithm is found below.

Algorithm 1 Ramer-Douglas-Peucker pseudocode.

```
1: function RamerDouglasPeucker(P, ε, p_i, p_j)        ▷ P = ⟨p_1, ..., p_n⟩
2:     Find vertex p_f ∈ ⟨p_i, ..., p_j⟩ farthest from the line p_i p_j.
3:     dist ← distance between p_f and p_i p_j.
4:     if dist < ε then
5:         RamerDouglasPeucker(P, p_i, p_f)
6:         RamerDouglasPeucker(P, p_f, p_j)
7:     end if
8:     Return line p_i p_j.
9: end function
```

The time complexity is $T(n) = O(n) + T(n-1) = O(n^2)$ in the worst case, but it is usually much faster in practice.

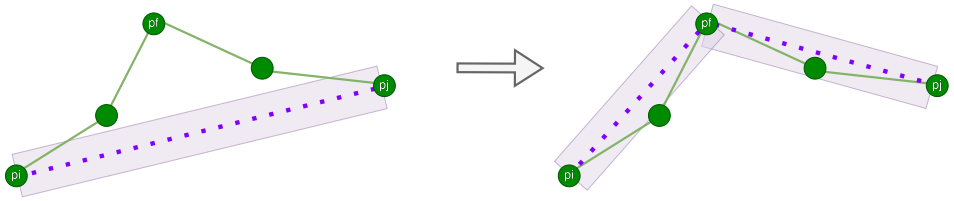

Figure 3.2: Example of one step Ramer-Douglas-Peucker simplification.

Driemel-HarPeled-Wenk is not recursive, unlike the previous, but instead simplifies the line starting from the very first point and iteratively removes points that fall within a certain radius from the current one, substituting all of them with a single line

that goes from the current point to the first point in the trajectory more distant than that radius. The pseudocode is below.

Algorithm 2 Driemel-HarPeled-Wenk pseudocode.

```
1: function DriemelHarPeledWenk(P, ε)                    ▷ P = ⟨p_1, ..., p_n⟩
2:     P' ← p_1.
3:     i ← 1.
4:     while i ≤ n do
5:         curr ← p_i.
6:         j ← i + 1.
7:         while dist(p_i, p_j ≤ ε) do
8:             j ← j + 1.
9:             if j > n then
10:                i ← n.
11:                Break.
12:            end if
13:        end while
14:        Add p_j to P'.
15:    end while
16:    Return P'.
17: end function
```

Either way, simplification has the side effect of modifying average speed estimates, as simpler, straighter paths imply a slower average speed compared to more complex, chaotic ones.

3.4 Stop Detection for Trajectory Segmentation

GPS data is normally formed by a continuous stream of points, but there is no explicit separation between the trajectories of the same entity. **Stop detection** is a technique used to identify moments in which the entity is staying still (or moving very slowly) to be used as endpoints of different trajectories. For example, if we consider a person moving through a regular work day, we may observe that person move away from a spot (likely his/her house), then stop at what is either school or work, then make another stop at the gym, and finally return back home. Every movement between each of these stops can be isolated as its own trajectory.

The general criteria is: the estimated speed from a set of points is very low (i.e., very little movement over a long time interval), the set of points represent a stop.

The sequence of points between stops is a trajectory. To check if a point belongs to a stop, typical space and time thresholds are $Th_S \in [50, 250]$ (meters), and $Th_T \in [1, 20]$ (minutes), but depending on the context, these values may change: thresholds for a car are different from those for a pedestrian, for example.

When it comes to vehicles specifically, stops can often be detected by simply checking when the engine is on or off. This is generally good, but may not detect stops that are very short. A more general approach is to use a density-based approach: the faster an entity is, the less of its own points are found around it as it is moving.

After trajectory segmentation has been performed, we can use them to study the behaviour of the individual, and eventually find deviations from the norm. A way to represent this information is building the entity's **Individual Mobility Graph**, where each location is a node and each movement is an edge; this means that for each trajectory, only the starting and ending locations are recorded in the graph. Naturally, denser areas with more nodes will represent important places. These locations are identified using a clustering algorithm, and are each associated with their frequency; trips between points belonging to the same cluster (location) are then aggregated as edges between the entire clusters. The basic assumption is that the location with the highest frequency is home, the second most frequent location is work, and so on. An alternative approach also considers the actual time of the day the traces were recorded at: points recorded during nighttime or early morning are more likely to be home, while those recorded during the morning and early afternoon are more likely to be work.

Other than GPS, phone data can also be used to infer important locations. By analyzing the calls and messages sent or received by the user, we can identify what are called **personal anchor points**, which are the most frequently contacted antennas. These points can also be labeled as home/work by considering the time of the day the calls were made or received.

3.5 Activity Labeling

After locations have been identified, an ulterior step is to assign to each of them a specific activity the user carries out once it arrives there. The first step is to assign a location to a set of actual point of interest that are close by. The POIs are then also associated with an activity, for example: "eating out", "shopping", "working", "education". At the end, the available data will be:

- For each POI, the category, the opening hours, the location;
- For each trajectory, the stop location, the stop duration, the time of day;

- For each user, the max walking distance (as in, how far the person is likely to walk after parking).

Each POI is associated to a certain probability of being the reason for the stop, calculated as a function of the information above. Finally, a stop is annotated with the activities and relative probabilities of the POIs close by.

Appendix A

Useful Tools and Libraries

A.1 File Formats

Shapefile Popular geospatial vector data format for GIS software. It can describe any vector feature using points, lines, and polygons. The format is actually composed of multiple files, but the shapefile itself has the filename extension `.shp`.

Specifies axis coordinates assuming a Cartesian coordinate system, using the ordering (x, y): x is longitude, and y is latitude.

A.2 Python Libraries

Shapely Used to analyze and perform set operations on planar geometric objects. Represents objects as either `Points`, `LineStrings`, or `Polygons`.

- `Points` can be 2- or 3-dimensional, and is defined from a coordinate tuple.
- `LineString` is built from a list of at least two coordinate tuples.
- `Polygon` is a filled area; must be defined by at least three coordinate tuples that speciy the outer perimeter, and optionally a list of inner holes.

Various attributes can be directly accessed from an instantiated object to analyze them (e.g., length, centroid, area, bounding box, etc.). For each of these types, a `Multi-` variant exists, used to have collections of objects of the same type.

GeoPandas Extends the datatypes used by pandas to allow spatial operation on geometric types. It's built on top of Shapely for geometric opeations. Its main data structures are `GeoSeries` and `GeoDataFrame`, which respectively extend pandas' `Series` ad `DataFrame` types. One constraint is that a `GeoDataFrame` must contain a `geometry` column that specifies the geometry of the objects held in the dataframe; this column must

be of type `GeoSeries`. The `.plot()` function (built on top of matplotlib's `.plot()`) can be used to easily visualize geometric data on a 2D map. Supports the shapefile data format, with methods for reading (`read_file()`) and writing (`to_file()`) shapefiles.

The main operations that can be performed on `GeoDataFrames` are spatial joins: the method used is `.sjoin()`, where parameter `how` accepts a string that specifies the type: `"left"`, `"right"`, or `"inner"`. The direction refers to the dataframe on which the method is called; for example:

```
df1.sjoin(df2, how="left")
```

performs a join that preserves all the rows of `df1`.

Scipy General library for scientific computing. Contains the module `scipy.spatial` that provides spatial algorithms and data structures. Provides a `cKDTree` class that implements efficient methods for nearest-neighbor calculations in spatial data.

Scikit-Learn The most popular machine learning library in Python.
Its `KNeighborRegressor` class can be used to perform k-NN regression on spatial data.

Scikit-Mobility Library for human mobility analysis. The library manages mobility data of various formats (CDR, GPS, social media data, etc.), and can represent trajectories and mobility flows using the data structures `TrajDataFrame` adn `FlowDataFrame`, respectively.

NetworkX Package for creation and manipulation of complex networks. Has implementations of many graph algorithms. Useful to operate on street networks, since they are usually represented as graphs with arbitrary attributes on nodes/edges.

OSMnx Package used to download, model, and visualize street networks from OpenStreetMap. It can retrieve walking, driving, and biking networks, as well as points of interests, transit stops, street orientations, travel time, elevation data, and routing. It contains several methods that automatically download data by providing some spatial reference; for example, `graph_from_address()` downloads and returns the desired type of network within some distance to a specific provided address; `graph_from_point()` is similar to the previous, but downloads the network centered around a point expressed as a tuple *(lat, lon)*; `graph_from_bbox()` allows the user to specify a bounding box to download the network within.

Geovoronoi Can create and plot voronoi tessellations. Has a main function called `voronoi_regions_from_coords()`, which takes as input a list of coordinates representing the centroids of the cells and an area to bound the cells, and returns a list of polygons representing the tessellation. It uses Scipy to perform the tessellation, and Shapely to cut the edge polygons to fit into the provided shape (since they would be otherwise infinite).

PySal Provides various methods for spatial analysis. It has four main modules:

- `Lib` contains the core spatial data structure and file IO functionalities;
- `Explore` contains methods for exploratory analisys. It contains the subpackage `esda`, which provides specific methods for spatial autocorrelation (Moran's I, Geary's C);
- `Model` contains different models to estimate spatial relationships, both linear and non-linear;
- `Viz` contains visualization tools.

Pykrige Implements various kriging algorithms, both for 2D and 3D data.

A.3 Links

opencellid.org - World's largest open database of cell towers.

Bibliography

[1] Kang-Tsung Chang. *Introduction to geographic information systems*. Mcgraw-hill Boston, 2018.

[2] Manuel Gimond. Intro to gis and spatial analysis. `https://mgimond.github.io/Spatial/`.