“sensormap extension”

Hausübung 3

May 6, 2019

1 Datensatz

In diesem Assignment erweitern Sie das letzte Assignment um einen neuen Sensortyp und um aggregierte Daten auf der Karte direkt dazustellen.

Verwenden Sie die eine getrimmte Version der Angabedaten vom letzten Assignment.[1]. Dieses Assignment beschränkt sich auf den Raum Graz.

2 Tasks

2.1 Offizielle Messwerte der Stadt Graz und Luftgüte (4p)

2.1.1 Aufbereiten und Einlesen der offiziellen Sensoren

Die Stadt Graz veröffentlicht Luftgütedaten von sechs Sensoren aus dem Raum Graz.[2]. Die Koordinaten der Stationen sind den offiziellen Luftgüteberichten zu entnehmen.[3]. Verwenden Sie die Detailsuche unter http://app.luis.steiermark.at/luft2/suche.php um die entsprechenden Daten auf Ihr System zu laden. Verwenden Sie dafür folgende Konfigurationen: (i) Alle Sensoren, deren Namen mit *Graz-* beginnen **und** die Komponente PM10 besitzen. (PM2.5 wird in diesem Task nicht benötigt). (ii) Beschränken Sie den Zeitrauf auf die zeitlich ersten und letzten Werte Ihrer bisherigen Daten (von luftdaten.info). Extrahieren Sie die Daten (evtl per Hand) aus den .xls Dateien und erstellen die Inputdateien. Verwenden Sie die Koordinaten aus dem Jahresbericht um die Position der Sensoren zu definieren. Speichern Sie die aufbereiteten Daten in einem Ordner namens `data` und fügen Sie diesen Ihrer Abgabe hinzu!

2.1.2 Luftgüte

Berechnen Sie die relative Luftqualität für jeden Sensor. Berechnen Sie dafür die durchschnittlichen Werte der Stadt. Danach berechnen Sie für jeden Sensor die Abweichung zu diesem Durchschnitt. Der schlechteste Sensor (höchste negative Abweichung) soll den Wert -1 und der beste Sensor (beste positive Abweichung) soll den Wert $+1$ haben. Alle anderen Werte sollen proportional zu diesen Limits sein. Speichern Sie dies in eine Property namens `air_quality`.

Wichtig: Führen Sie diese Berechnung getrennt für die offiziellen und privaten Sensoren durch!

[1] https://github.com/pkasper/info2-bm/blob/master/2019/assignments/assignment_3/sensors_ass_3.csv
[2] http://www.umwelt.steiermark.at/cms/ziel/3611708/DE/
[3] Seite 56ff. http://app.luis.steiermark.at/berichte/Download/Jahresberichte/Jahresbericht_2017_C.pdf

2.1.3 Package Anpassen

Passen Sie ihr Package aus dem letzten Assignment an. Folgende Änderungen sind notwendig:

- Die Klasse `Sensor` soll die readonly Property `official` erhalten. Diese hat den Wert `False` bei allen alten Daten und `True` bei den neuen offiziellen Sensordaten.
- Die Funktion `parse_official(data_path)` (in `sensormap_functions`). Diese Funktion liest die von Ihnen aufbereiteten Daten der offiziellen Sensoren ein.
- Die Funktion `create_bokeh_plot` soll überprüfen, ob Werte für PM2 existieren (diese fehlen ja bei den offiziellen Sensoren). Plotten Sie PM2 nur, wenn es auch Werte gibt.
- Die Klasse `Sensor` soll die Property `air_quality` besitzen. Zulässige Werte sind Floats von -1 bis $+1$.

2.2 Voronoi Diagramm (6pt)

Fügen Sie der `City` Klasse eine Funktion `calculate_voronoi(official)` hinzu. Diese Berechnet ein Voronoi-Diagramm aus allen Sensoren der Stadt. Der Parameter `official` dient dazu, um zu definieren, ob das Diagramm über die offiziellen oder privaten Sensoren berechnet werden soll.

Verwenden Sie zur Implementierung die `Voronoi` Funktion von Scipy.[4]. Setzen Sie dabei den Parameter `qhull_options = "Qc"`.

Passen Sie die Klasse `Sensor` an. Geben Sie Ihr eine Property namens `vor_region`, welche per Default den Wert `None` hat. Der Setter soll lediglich `None` oder Listen (oder numpy Arrays) zulassen.

Die beiden folgenden Subtasks sind aufbauend. Implementieren Sie erst die Basic Variante und erweitern Sie diese danach um die unendlichen Regionen.

2.2.1 Basic

Berechnen Sie die Voronoi Regionen der Sensoren. Das Rückgabeobjekt der Funktion hat eine Reihe von Properties. `point_region` bietet Ihnen die Möglichkeit die Region mit einem Input zu assoziieren. Dadurch können Sie die Region wieder einem Sensor zuordnen. `regions` assoziiert die Punkte zu Vertices (konkreten Koordinaten). Sollte eine Region einen Wert `-1` besitzen, dann ignorieren Sie diese Region in diesem Schritt. **Hinweis:** Es gibt auch immer eine leere Region. Diese ist ebenfalls zu ignorieren. Speichern Sie die Koordinaten der Region (die Vertices) in die Property des jeweiligen Sensors.

2.2.2 Infinite Regions

Berechnen Sie nun die Vertices der unendlichen Regionen. Die Vorgangsweise ist wie folgt: Iterieren Sie über die Werte der Property `ridge_vertices` (am besten mit enumerate). Wennimmer ein Wert -1 ist, dann geht diese Kante ins unendliche. Es ist immer zumindest ein Punkt bekannt ($\neq -1$). Um die Kante selbst zu berechnen extrahieren Sie die Punkte der Kante (z.B., via `vor_diag.points[vor_diag.ridge_points[i]`; `i` ist hier der Index des ridge_vertex.) Von diesen beiden Punkten berechnen Sie den Mittelwert (via `np.array.mean(axis=0)`). Wählen Sie die Richtung des Vektors, sodass dieser sich vom Zentrum aller Punkte wegbewegt. Multiplizieren Sie den Vektor so, dass er die Länge 0.3 hat. Speichern Sie nun den neuen Punkt in einem dict mit Key `tuple(-1, <index_des_bekannten_punktes>)` und dem neuen Punkt als Wert (bekannter Punkt + Vektor).

Wennimmer Sie nun beim Iterieren über die Regionen auf -1 treffen, dann behandeln Sie dies wie folgt: Überprüfen Sie, ob Ihr Punkt einen linken und rechten Nachbar hat. Ist ein linker Nachbar vorhanden (also Index des Punktes > 1), dann fügen Sie den Punkt aus dem vorhin erstellten dictionary mit dem Key (-1, index-1) ein. Selbiges machen Sie für den rechten Nachbarn, wenn index+1 existiert (nur diesmal mit dem Key (-1, index+1)).

Hinweis: Schreiben Sie sich eine Hilfsfunktion, welche die Länge eines Vektors berechnet. **Hinweis:** Die SciPy Funktion `scipy.spatial.voronoi_plot_2d` nimmt ein Objekt der Voronoi Klasse entgegen und Zeichnet die Regionen. Verwenden Sie zum, um Ihr Ergebnis zu vergleichen.

[4] https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.spatial.Voronoi.html

2.3 Plotting (2p)

2.3.1 Layers in Folium

In diesem Task müssen Sie die `create_map` Funktion aus Ihrer letzten Abgabe anpassen. Im ersten Schritt sollen Sie die Layer von Folium verwenden. Eine Instanz legen Sie mit `folium.map.FeatureGroup(<name>)` an. Führen Sie danach die einzelnen Komponenten dem Layer hinzu (via `layer.add_child(new_component)`. Führen Sie danach den Layer zur Karte hinzu (via `folium_map.add_child(layer_item)`). Schlussendlich fügen Sie der Karte via `folium_map.add_child(folium.map.LayerControl())` noch ein Kontrollobjekt hinzu. Dadurch wird am rechten oberen Rand eine Box gezeichnet mit welcher der Benutzer die einzelnen Layer ein- und ausblenden kann.

Folgende Layer sind zu implementieren:

Sensors Official - Die offiziellen Sensoren der Stadt Graz.

Sensors Private - Die inoffiziellen Sensoren (Ihre bisherigen Daten).

Overlay Official - Das Voronoi Overlay aus allen offiziellen Sensoren.

Overlay Private - Das Voronoi Overlay aus allen privaten Sensoren.

Hinweis: Es ist für dieses Assignment ausreichend, wenn Sie im Parameter `plot_list` nur die Stadt Graz übergeben, da wir nur hier offizielle Sensoren besitzen.

2.3.2 Voronoi Overlay

Verwenden Sie die `vor_region` der `Sensor` Klasse. Zeichnen Sie für jeden Sensor die entsprechende Region auf die Karte. Die Farbe der Region soll dem Wert der Property `air_quality` auf einer Skala von Rot (schlechteste Region) nach Grün (beste Region) entsprechen. Verwenden Sie dabei auch einen `alpha` Wert um das Overlay ein wenig durchsichtig zu machen. Verwenden Sie zum Zeichnen der Voronoi-Regionen die Funktion `folium.Polygon`.

Hinweis: Farbcodes des Polygons sind Hexcodes (z.B., `#FF0000` für rot). Setzen Sie sowohl das Keyword-Argument `color`, als auch `fill_color`. Zur Kontrolle der Durchsichtigkeit gibt es den Parameter `fill_alpha`.

Hinweis: Einzelne Regionen werden sehr groß und reichen bis weit über die Stadtgrenzen hinaus. Dies ist eine Limitation des Algorithmus und vollkommen erlaubt!

2.4 Testen (4Pt)

In der Datei `assignment_3.py` sollen Sie nun Ihr Package verwenden. Testen Sie die Funktionalität Ihres erweiterten Packages. Beschreiben Sie Ihre Überlegungen und Vorgangsweise sowohl in Codekommentaren als auch im Report. Testen Sie unter Anderem auch, was passiert, wenn Sie nicht zulässige Werte für Parameter verwenden.

2.5 Report (4p)

Schreiben Sie einen Report, welcher Ihre Arbeit und Ihre Erkenntnisse beschreibt. Fügen Sie den Report als PDF-Datei Ihrer Abgabe hinzu. Folgende Komponenten sind hierbei wichtig:

- Wie Sie die einzelnen Arbeitsschritte auf die Teammitglieder aufgeteilt haben.
- Wie Sie die offiziellen Daten aufbereitet haben.
- Auf welche Probleme Sie gestoßen sind.
- Wie Sie Ihr Programm getestet haben.
- Interpretation der Ergebnisse.
  - Verlässlichkeit der Sensorwerte.
  - Unterschiede zwischen offiziellen und privaten Messwerten.
  - Alles, was Ihnen sonst noch auffällt.

3 Bonus: Kombination naheliegender Sensoren (5p)

Bei diesem Task sollen Sie Sensoren, die nahe beieinander liegen, kombinieren. Schreiben Sie hierfür eine Funktion `combine_sensors(<int>)` inder `City` Klasse. Der Parameter ist der Minimalabstand zwischen Sensoren. Also alle unter diesem Wert sollen kombiniert werden. Achten Sie darauf, dass Ihre Implementation die beiden Sensortypen getrennt voneinander behandelt (offiziell vs. luftdaten.info). Wie Sie die Sensoren kombinieren, bleibt Ihnen überlassen.

Folgende Überlegungen sind erforderlich:

- Wie kombinieren Sie die Sensorwerte?
- Was ist die Position des kombinierten Sensors?
- Was passiert, wenn ein Sensor mehr als nur einen nahen Nachbar hat?

Beschreiben Sie Ihre Überlegungen, Vorgangsweise und Ergebnisse auch im Report!
Wichtig: Es gibt hier keine völlig falschen Lösungen.

4 Beschränkungen

- Fügen Sie keine nutzlosen Komponenten Ihrer Abgabe hinzu
- Sofern möglich, verwenden Sie eingebaute Funktionen
- Importieren Sie nur erlaubte Packages und solche, die Sie dann auch verwenden
- Verwenden Sie keine Kommandozeilenparameter

4.1 Erlaubte Packages

- `os`, `sys`, `math`
- `numpy`, `scipy`, `pandas`, `tqdm`
- `matplotlib`, `seaborn`, `bokeh`, `folium`, `geopy`

5 Datei Header

All Ihre Quelldateien (oder Notebooks) in Ihrer Abgabe müssen gleich zu Beginn einen Kommentar mit folgenden Informationen enthalten:

- Author: – Ihr Name
- MatNr: – Ihre Matrikelnummer
- Description: – Generelle Beschreibung der Datei
- Comments: – Kommentare, Erklärungen, usw

Bitte einfach den folgenden Code in Ihre Dateien kopieren und den Inhalt anpassen. Je nach PDF-Reader müssen Sie eventuell die Leerzeichen/Einrückungen per Hand anpassen. Bei jupyter Notebooks fügen Sie den Kommentarheader bitte in die erste Zelle ein.

Beispielheader:

```
########################################################################
# Author:      [FIRST NAME] [LAST NAME]
# MatNr:       [MATR NR]
# Description: [SHORTDESCRIPTION]
# Comments:    [ANY RELEVANT COMMENTS.
#              CAN BE MULTILINE]
########################################################################
```

6 Coding Standard (Abzug bis 50%)

Für diese Lehrveranstaltung orientieren Sie sich am offiziellen PEP 8 Standard[5]. Dieser Beschreibt grundsätzliche Formalitäten im Bezug auf Ihren Code. Folgendes ist besonders zu Beachten:

Sprache. Code schreibt man in Englisch. Im internationalen Zeitalter ist es notwendig, dass auch jemand am anderen Ende der Welt verstehen kann, was Sie programmiert haben. Ihr gesamter Quellcode muss daher auf Englisch geschrieben sein. Dies betrifft sowohl die Kommentare also auch Variablennamen und Ähnliches.

Leerzeichen statt Tabulatoren. Python basiert auf Einrückungen, anstatt auf geschwungenen Klammern . Theoretisch gibt es die Möglichkeit, Leerzeichen (spaces) oder Tabulatoren (tabs) zu verwenden. PEP8 schreibt aber klar 4 Leerzeichen als Einrückungen vor. Die meisten Python Programmierumgebungen werden automatisch 4 Leerzeichen einfügen, wenn Sie auf die Tabulator-Taste drücken.

Sprechende Namen. Verwenden Sie kurze, aber sprechende Namen für Ihre Variablen, Funktionen, (und Ähnliches). Es muss eindeutig aus dem Namen hervorgehen, was die Aufgabe des Elements ist. Für simple Iterationen kann ein einfaches i ausreichend sein, aber dies kann schnell zu Chaos führen. Sollten Sie Variablen haben, die keine Aufgabe haben und nicht verwendet werden, schreibt PEP 8 vor, einen einfachen Unterstrich (_) zu verwenden. Achten Sie bei der Groß-Kleinschreibung auf die Richtlinien aus PEP8.

120 **Zeichen Zeilenlänge.** PEP8 sieht Zeilenlängen von maximal 79 Zeichen vor. Ein anderes, etwas großzügigeres Limit, welches sich in der Szene etabliert hat, sieht eine Länge von 120 Zeichen vor. In dieser LV verwenden wir daher das erweiterte Limit. Bitte achten Sie darauf, dass keine Zeile in Ihrem Code diese Länge von 120 Zeichen überschreitet (gilt auch für Kommentare). *Hinweis:* Zeilenlänge inkludiert die Einrückungen mittels Leerzeichen!

7 Automatisierte Tests

Ihr Programm wird automatisiert getestet. Achten Sie daher, dass Sie die Angabe genau einhalten. Dies gilt im Besonderen für vorgeschriebene Variablen- und Funktionsnamen!
Vergewissern Sie sich, dass Ihre Abgabe allen Beschränkungen, die in dieser Angabe erwähnt sind, konform ist. Zusätzlich wird jede Abgabe auch von einem Mitglied des Tutorenteams begutachtet. Ihre Tutoren führen auch die finale Bewertung (Punkte) durch.

[5] `https://www.python.org/dev/peps/pep-0008/`

8 Abgabe

8.1 Deadline

02. Juni 2019 um 23:59:59.

Eine Spätabgabe ist nicht vorgesehen. Ausnahme bilden hier Notfälle. Sollte das Abgabesystem nicht online sein, verlängert sich die Deadline automatisch um 24 Stunden. Sollten Sie, aus diversen Gründen, nicht in der Lage sein, Ihre Abgabe hochzuladen, kontaktieren Sie Ihren Tutor *VOR* der Deadline. (*Hinweis:* Urlaub oder Ähnliches wird nicht als Grund akzeptiert!)

8.2 Hochladen der Abgaben

Assignments werden stets als Archive abgegeben. Erlaubt sind hier die Formate *.zip*, und *.tar.gz*.
Zusätzlich zu ihren Quelldateien, soll Ihre Abgabe auf eine Datei namens *readme.txt* beinhalten. Das Vorhandensein der Datei ist Pflicht, ihr Inhalt aber optional. Sie soll folgenden Inhalt haben: (i) Die Zeit, die Sie benötigt haben, um die Aufgabenstellung zu absolvieren. (ii) Feedback, wo Sie Probleme hatten. Ihr Feedback ermöglicht es, verbreitete Probleme zu erkennen und die Vorlesung und Tutoriumseinheiten entsprechend anzupassen.
Abgaben erfolgen auf der Palme Website. Bitte prüfen Sie vor der Abgabe diese Kriteren:

- Datei- und Ordnerstruktur (siehe unten)
- Kommentarheader in jeder Quelldatei
- Coding Standard

8.3 Struktur der Abgabe

```
assignment_3.zip (or assignment_3.tar.gz)
├── assignment_3.py
├── readme.txt
├── report.pdf
├── data
│   └── [Ihre aufbereiteten Daten der Offiziellen Sensoren]
└── sensormap
    ├── __init__.py
    └── [Alle Dateien Ihrer Abgabe(.py)]
```