3.lab_session

Joris Voogt, Student number: 4295978

September 19, 2023

Import relevant packages here.

[1]:
```
import math
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Load the data and verify it is loaded correctly.

Print it (head, tail, or specific rows, choose a sensible number of rows).

Compare it to the source file.

[2]:
```
data = pd.read_csv("cf_data.csv")
print(data.head())
```

```
         dv        s         a
0 -0.743240  53.5427  1.242570
1 -0.557230  53.6120  1.777920
2 -0.454769  53.6541  0.544107
3 -0.525396  53.7030 -0.294755
4 -0.601285  53.7592 -0.290961
```

In the ensuing, you will use numpy.

Let's create a grid for the values to plot. But first create two arrays named dv and s using numpy.linspace that hold the grid values at the relevant indices in their respective dimension of the grid.

Create a grid named a with zeros using numpy.zeros in to which calculated acceleration values can be stored. Let the grid span:

Speed difference dv [m/s]

From -10 till 10

With 41 evenly spaced values

```
</li>
<li>Headway <code>s</code> [m]
    <ul>
        <li>From 0 till 200</li>
        <li>With 21 evenly spaced values</li>
    </ul>
```


[3]:
```
dv = np.linspace(-10, 10, 41)
s = np.linspace(0, 200, 21)
a = np.zeros([21, 41])
```

Create from the imported data 3 separate numpy arrays for each column dv, s and a. (We do this for speed reasons later.)

Make sure to name them differently from the arrays that belong to the grid as above.

You can access the data of each column in a DataFrame using data.xxx where xxx is the column name (not as a string).

Use the method to_numpy() to convert a column to a numpy array.

[4]:
```
DV = data.dv.to_numpy()
S = data.s.to_numpy()
A = data.a.to_numpy()
```

Create an algorithm that calculates all the acceleration values and stores them in the grid. The algorithm is described visually in the last part of the lecture. At each grid point, it calculates a weighted mean of all measurements. The weights are given by an exponential function, based on the 'distance' between the grid point, and the measurement values of dv and s. To get you started, how many for-loops do you need? For this you will need math. Use an upsilon of 1.5m/s and a sigma of 30m. Warning: This calculation may take some time. So:

Print a line for each iteration of the outer-most for-loop that shows you the progress.

Test you code by running it only on the first 50 measurements of the data.

The following code will plot the data for you. Does it make sense when considering:

Negative (slower than leader) and positive (faster than leader) speed differences?

Small and large headways?

[5]:
```
# ...
upsilon = 1.5
sigma = 30

# Calculates weight of x and y dimensions using upsilon and sigma.
def weight(x_val, x_grid, y_val, y_grid):
    return math.exp(-(abs(x_val - x_grid) / upsilon) - (abs(y_val - y_grid) /␣
 ↪sigma))

# Calculates the weighted average of the third column in the data variable.
def weighted_avg(grid, x_ax, y_ax, data):
    for row in range(len(grid)):
        for column in range(len(grid[0])):
            sum_w, sum_a = 0, 0
```

```
            for x, y, z in data:
                w = weight(x, x_ax[column], y, y_ax[row])
                sum_w += w
                sum_a += w * z

            grid[row, column] = sum_a / sum_w

    return grid

a = weighted_avg(a, dv, s, list(zip(DV, S, A)))
```

[6]:
```
X, Y = np.meshgrid(dv, s)
axs = plt.axes()
p = axs.pcolor(X, Y, a, shading='nearest')
axs.set_title('Acceleration [m/s/s]')
axs.set_xlabel('Speed difference [m/s]')
axs.set_ylabel('Headway [m]')
axs.figure.colorbar(p)
axs.figure.set_size_inches(10, 7)
```