My current research focuses on Mobility Modelling and Analytics, Mobility Data mining, Mobile Phone Cellular Network data Analytics and Intelligent Transportation Systems (ITS).

ITS Lab research activities are covering four main directions:

• Mobility Analytics
• Advanced Travel Information Systems
• Advanced Driver Assistance Systems
• Vehicular Networking

ITS Lab: its.cs.ut.ee
homepage: http://kodu.ut.ee/~hadachi/HomePage.html
Syllabus

• Lecturer: Amnir Hadachi
• Lab Assistant: Artjom Lind
• Course webpage: https://courses.cs.ut.ee/2022/DS4UM/spring

• Lectures:
  • Wed. 10h15 -12h00
    • Live stream / Recording
  • Fri. 10h15-12h00
    • Live stream
Syllabus

• Two assignments (50%)

• Final exam (50%)

• Labs:
  • Lab sessions will be in format of practice and tutorials
  • Some of the lab session will include exercises that we will be solve during the session or for take home practice.
## Syllabus

<table>
<thead>
<tr>
<th>Block</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction: Data Science &amp; Big data</td>
</tr>
<tr>
<td>2</td>
<td>Databases and Data Preparation</td>
</tr>
<tr>
<td>3</td>
<td>Spatial Data</td>
</tr>
<tr>
<td>4</td>
<td>Bayesian Inference</td>
</tr>
<tr>
<td>5</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>6</td>
<td>Spatial Analysis</td>
</tr>
<tr>
<td>7</td>
<td>Data Visualization</td>
</tr>
<tr>
<td>8</td>
<td>Professional issues and ethics</td>
</tr>
</tbody>
</table>
Part 1:
“Data Science” and “Big Data”
Introduction

Modern History

Important dates for Data Science

2012
2011
1996
2008

“The Sexiest Job of the 21st century

“Predicted 1.5M job demand for data scientist

“Used in its current meaning

“Used as a Job title
Introduction

- E-commerce
- Banking
- Manufacturing
- Data Science
- Applications
- Transport
- Healthcare
- Finance
Introduction

Data Science Applications

Transport

- OD-Matrix Analysis
- Airline Pricing and arbitrage
- Pothole Monitoring
- Social review apps
- Self-driving cars
- Taxi services
Examples

Origin-Destination Analysis

• Objective: Estimating origin and destination of the actual routes taken by the drivers.

• Input data: Automatic Number Plate Recognition and Induction loop.

• DS role: Applying Data fusion and data science for recovering missing or corrupted data.

• Side-effect: tracking and identifying individuals trajectories.

Figure 10: Typical origin-destination profiles.

Examples

Origin-Destination Analysis

- **Objective:** Depicting large scale of the population commuting
- **Input data:** Mobile network data
- **DS role:** Data enrichment and extracting the OD-matrix

Examples
Airline Pricing and arbitrage

- Objective: Dynamic pricing
- Input data: demand over time.
- DS role: isolate the affecting factors such as travel type (Business/Leisure), demographic, wealth of the customers.
- Advanced: building clients profile and target marketing.
Examples
Pothole Monitoring

- Objective: detecting pothole in asphalt pavement.
- Input data: images or social app
- DS role: decision support for classification and maintenance scheduling.

Examples

Social review apps

- Objective: Providing review and recommendation of public places and restaurants in a sociable manner.

- Input data: location and activity labels.

- DS role: building an understanding of the urban mobility and the motivation behind it.
Examples
Self-driving cars

• Objective: dependent on data

• Input data: different sensors from the embedded system in the car, sensors and infrastructure.

• DS role: making sense of this data.
Examples
Self-driving cars - NetCalib

- **Objective**: Auto-calibration of Lidar and Stereo Camera
- **Input data**: Lidar and Stereo Camera
Examples
Taxi services

• Objective: providing manual and self-driving taxi service where data is collected about the passengers and their journeys.

• Input data: many…

• DS role: is to enhanced the existing system and providing a more customised services. Building a service around the needs of the citizens, providing an understanding of the public demand for transport, or sell the data to insurance companies.
Remark 1.1:

All these applications presented has something in common that makes them “Data Science” or “Big Data”, according to Mayer-Schonberger and Cukier that the fundamental changes in the scientific approaches is about three points:

* **Bigness**: The ability to analyse massive amount of data at once about a topic rather than sampling a smaller set.

* **Messiness**: The power of skipping cleaning, data preprocessing and structuring and execute algorithms directly on raw data.

* **Correlation**: discarding the quest for elusive causality and focusing on correlation.
Terminology

**Definition 1.1:** Data Science

Data Science is defined as an inter-disciplinary field that applies scientific methods, processes, algorithms, systems to discover and extract knowledge from structured or unstructured data.

NB: We emphasise that data science is about the use and re-use of data collected passively rather than via causal experiments used in regular “science”.

**Definition 1.1:** Big Data

We say that we are dealing with Big Data the moment our traditional single computer processor is incapable of handling the data and there is a need for using parallel computation or distributed.
How to make your CV reflect that you are a data scientist?

Answer: In general the term itself is a fuzzy concept that can refer to many characteristics which can label it in a rough, overlapping and clusters of idea that can be listed as follows:
Remark 1.1: Ethics

How the data collected to used and to which end? Where is the limit or the border line?
Data Scientist

Data Science General Workflow:

- Data Collection
- Exploration
- Data Manipulation
- Modelling
- Validation
- Reporting
Data Scientist

Data Varieties:

• Structured data

Definition 1.2: Structured Data

Structured data is highly organized data categorized as quantitative, and it is stored, proceeded, and manipulated in a traditional relational (SQL) database.
Data Scientist

Data Varieties:

- Structured data
- Unstructured data

**Definition 1.3: Unstructured Data**

Unstructured data does not have a predefined data model, making them unfit for structured database format. It is qualitative and usually generated from human activities.
Data Scientist

Data Varieties:

- Structured data
- Unstructured data
- Semi-structured data

**Definition 1.4: Semi-structured Data**

Semi-structured data does not fit in a structured database; however, it is still structured by tags that can be used to create a hierarchy in the data.
Puzzle:

Data Scientist

Data Collection / querying / consuming:

• Query data

• Common usage:
  • CSV
  • Scripting
  • App files
  • Web programming files
Data Scientist

Modelling:

• Mathematical modeling

• Statistical methods for insight discovery
Data Scientist

Modelling:

• Mathematical modeling
• Statistical methods for insight discovery

Coding, coding, and coding
Part 2: Introducing Python
Python Basics

• Core

The tradition when introducing a new language is to do a print Hello World.

```python
In [ ]: print('Hello World')

You can also print digits

In [ ]: print(2+4)

Defining a variable.

In [4]: x = 3

In [ ]: print(x+1)

String in python.

In [ ]: s='hello'+'world'

In [ ]: print(s)

To see the type of variable.

In [ ]: type(s)
```
Python Basics

- Lists

Lists

A list is about storing an order sequence of other variable and it is a type of variable.

```python
In [ ]: L=[1,2,3,'Kala',8.4,0.9,[7,3]]
```

```python
In [ ]: print(L)
```

Selecting a subset and creating a new list.

```python
In [ ]: L[0:4]
```

The same applies for strings.

```python
In [ ]: s='where is the dataset'
```

```python
In [ ]: s[1]
```

```python
In [ ]: s[0:2]
```

Remember that python counts "indexes" and the order starts at 0 not at 1.

Adding element in a list.

```python
In [ ]: L.append('Science')
```

```python
In [ ]: L.remove(3)
```
Python Basics

- Dictionaries

Dictionaries or also known as "Hash Tables" are another basic type which is generalizing the idea behind lists.

For example, we can represent persons names and their associated licence plates:

```
In [ ]: d=dict()
In [ ]: d['Jack Williams']='234 THG'
In [ ]: d['Sandra Miller']='779 TLR'
```

They can be considered as a simple form of DB from a Data Science perspective. We will discover more about this later in other lectures.
Python Basics

• Control structure

Control Structures

Similarly to other programming languages, Python allows you to use loops and conditions.

In []: for i in range(0,8,4):
    print('Hello world'+str(i))

In []: x=1
    while x<10:
        x+=1 # equivalent to x=x+1
        print('while'+str(x))
    if x==1:
        print('loop started and x is one')
    else:
        print('x is not one')
Python Basics

• Files

For writing a file using python.

```
In [1]: f=open('myfilename.txt','w')
In [2]: f.write('hello')
In [3]: f.close()
```

For reading. (Of course there is other way which we will see later)

```
In [1]: for line in open('myfilename.txt'):
   ...:     print(line)
```
Python Basics

- Functions

We can define a function as a self-contained piece of code that takes an input and performs a task on it and gives an output.

```python
In [ ]:
def myfunction(x,y):  # x and y are my input
    result=x+y
    return result  # returning the output

a= 3
b= 4

z = myfunction(a,b)
print(z)
```

Python can also be used for object-oriented programming (OOP). This allows for complex data types such as classes to be defined with their functions.

When it is the case, the name of the function is appended to the name of the object.

```python
In [ ]:
myobject.myfunction(x,y)
```

In other words, this is equivalent to.

```python
In [ ]:
myfunction(myobject, x,y)
```
The moment your code starts to be more than one page long it is a sign to start splitting into different files or what is called Modules in Python (or libraries / components in other programming languages).

```python
Mymodule.py

```def myfunction(x,y): # x and y are my input
result=x+y
return result # returning the output

```from mymodule import *
z = myfunction(6,3)

Or

```from mymodule
z = mymodule.myfunction(6,3)
```
Python Basics

• Libraries

• Mathematics

`math — Mathematical functions`

This module provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the `cmath` module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don’t is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

**Number-theoretic and representation functions**

`math.fsum(iterable)`

Return an accurate floating point sum of values in the iterable. Avoids loss of precision by tracking multiple intermediate partial sums:

```python
>>> sum([.1, .1, .1, .1, .1, .1, .1, .1, .1])
0.9999999999999999
>>> fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1])
1.0
```

The algorithm’s accuracy depends on IEEE-754 arithmetic guarantees and the typical case where the rounding mode is half–even. On some non–Windows builds, the underlying C library uses extended precision addition and may occasionally double–round an intermediate sum causing it to be off in its least significant bit.

**Power and logarithmic functions**

`math.exp(x)`

Return e raised to the power x, minus 1. Here e is the base of natural logarithms. For small floats x, the subtraction in exp(x) - 1 can result in a significant loss of precision; the `expm1()` function provides a way to compute this quantity to full precision:

```python
>>> from math import exp, expm1
>>> exp(1e-5) - 1 # gives result accurate to 11 places
1.000005000069649e-05
>>> expm1(1e-5) # result accurate to full precision
1.000000000016668e-05
```
Python Basics

- Libraries
- Plotting
Python Basics

- Libraries
- Data Frames

### Data Frames

```python
In [1]: import pandas as pd  # import pandas library and refer to it as pd
In [1]: import numpy as np  # import numpy Library and refer to it as np
In [1]: data = pd.read_csv('dataRoadSafetyData_Accidents_2018.csv')  # read csv file using pandas function
In [1]: data.columns  # print name of columns
In [1]: data.shape  # print number of rows and columns
In [1]: data['Number_of_Vehicles']  # extract a the column with header name "Number_of_Vehicles"
In [1]: data[0:20]  # Extract some rows
In [1]: data.iloc[4]  # Extract single row with index 4
In [1]: data[::10, :]  # Extract every 10th row
In [1]: data.as_matrix()  # Convert to numpy DF
In [1]: I = np.eye(3)  # create a matrix
In [1]: data = pd.DataFrame(I, columns=['col1','col2','col3'])  # convert numpy DF to pandas DF
In [1]: data['col1'] = data['col2']  # add column
In [1]: data + data  # add all columns
In [1]: del data['col2']  # delete column
In [1]: data.append(data)  # append frame
In [1]: data.sort_values('col3', ascending=False)  # sort by column
In [1]: data['newField'] = 0  # add extra column
In [1]: data[data['col3'] == 0]  # select rows where condition is true
In [1]: data.merge(data)  # join columns from frame
In [1]: data = data.merge(data, how='outer').fillna(method='ffill')  # align by time
In [1]: data.to_csv('filename')  # save frame as csv file
```
Most common problem are related to data manipulation or glitches into data itself.
Additional Material

Please refer to Jupyter Notebook in the course webpage.

Python For Data Science - Initiation.ipynb