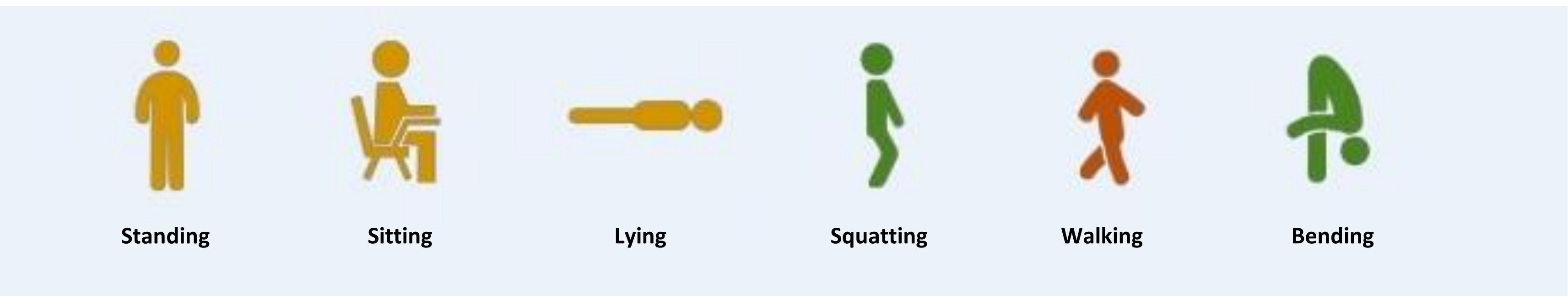
Data Science for Assistive Health Technologies

Dr. Muhammmad Adeel Nisar

Department of Information Technology
University of the Punjab, Lahore

Courtesy: MDS, IMI, Universität zu Lübeck, Germany

Activity Monitoring

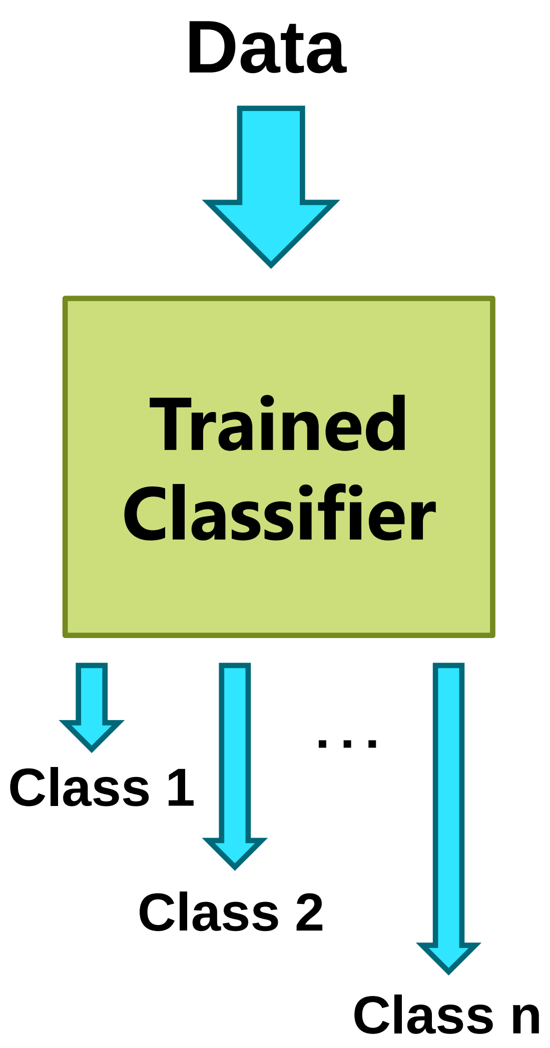


Contents

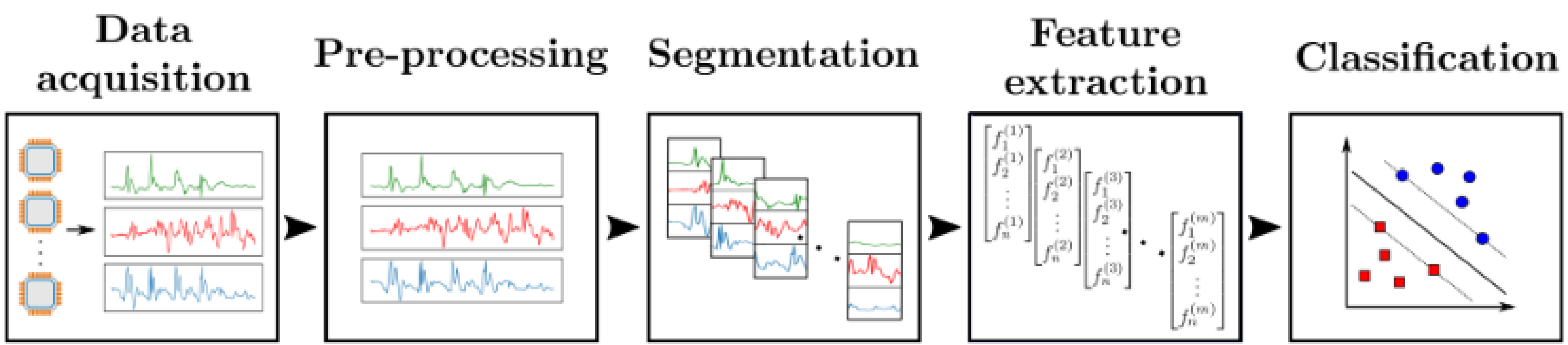
Project presentation and organisation

About activity monitoring

- Activity Monitoring is one popular application field of Ubiquitous Computing
- **Goal**: detect a specific set of activities using sensors worn by a subject providing data in real-time
- **Applications**: surveillance, gaming, remote control, assistive living, …
- **Principle:** classification problem
- **Problem:** how to obtain a (good) trained classifier?



Pattern Recognition Chain (PRC)



- Choice and setup of sensors
- Elaboration of experimental protocols
- Subject recruitment

- Noise removal
- Data normalisation
- Data downsampling

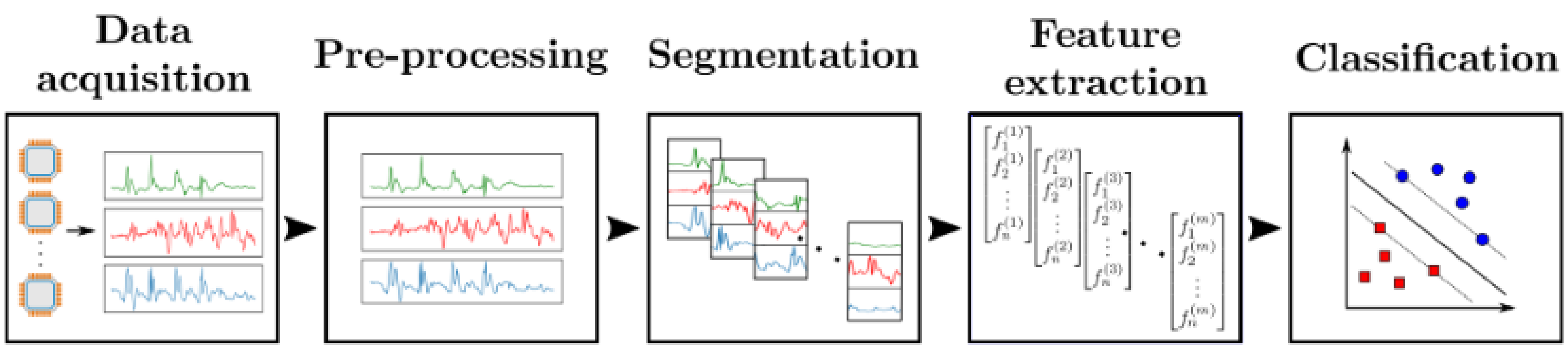
- Data dimensionality reduction
- Removal of data irrelevant to the recognition problem

- Data dimensionality reduction
- Computation of values relevant to the recognition problem

- Training of a classifier in the space of features
- Classifier evaluation

The Cognitive Village dataset (1/6)

- In practice, data acquisition step = one of the most difficult step of the processing chain
- In this project, we will use the **Cognitive Village (CogAge) dataset**
- **Note**: some steps of the PRC have already been processed with the CogAge dataset.



Already done | To be done in the project (?) | Already done | To be done in the project | To be done in the project

The Cognitive Village dataset (2/6)

(Activities with * are those for which left or right hand execution matters)

- **61 activities** divided in two categories:
  - **State:** 6 activities characterising the pose of an individual
  - **Behavioural:** 55 activities characterizing the behaviour of an individual
- 4 subjects
- Each activity repeated at least 20 times by each subject
- Each activity lasts for 4 seconds

State activities					
Standing	Sitting	Lying	Squatting	Walking	Bending

Behavioral activities		
Sit down	Stand up	Lie down
Get up	Squat down	Stand up from squatting
Open door*	Close door*	Open drawer*
Close drawer*	Open small box*	Close small box*
Open big box	Close big box	Open lid by rotation*
Close lid by rotation*	Open other lid*	Close other lid*
Open bag	Take from floor*	Put on floor*
Bring	Put on high position*	Take from high position*
Take out*	Eat small thing*	Drink*
Scoop and put*	Plug in*	Unplug*
Rotate*	Throw out*	Hang
Unhang	Wear jacket	Take off jacket
Read	Write*	Type*
Talk using telephone*	Touch smartphone screen*	Open tap water*
Close tap water*	Put from tap water*	Put from bottle*
Throw out water*	Gargle	Rub hands
Dry off hands by shake	Dry off hands	Press from top*
Press by grasp*	Press switch/button*	Clean surface*
Clean floor		

The Cognitive Village dataset (3/6)

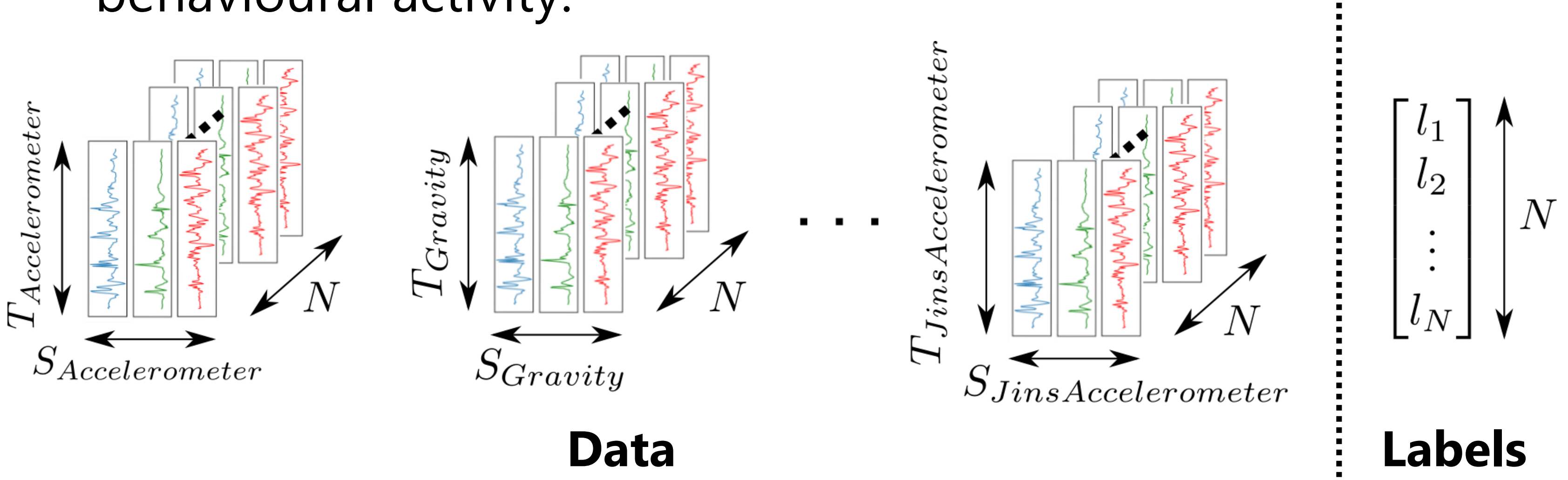
- Data acquisition performed using 3 wearable devices:
  - Google NEXUS 5X smartphone
  - Microsoft Band 2
  - JINS MEME glasses
- **Note 1**: each device has its own sampling rate
- **Note 2**: smartphone/smartwatch placed in the left pocket/on the left arm → differences between left- and right-hand executions for some behavioural activities
  - In this project, **executions using both hands** will be used

The Cognitive Village dataset (4/6)

- Dataset provided on this repository (executions for behavioural activities with both hands, total size ~159.3MB)
- Training and testing datasets + labels already prepared
- Data and labels are Numpy arrays (.npy files)
- One data file for each sensor from all 3 devices:
  - **Smartphone**: Accelerometer, Gravity, Gyroscope, LinearAcceleration (all sampled at 200Hz) + Magnetometer (50Hz)
  - **Smartwatch**: MSAccelerometer, MSGyroscope (67Hz)
  - **Smartglasses**: JinsAccelerometer, JinsGyroscope (20 Hz)

The Cognitive Village dataset (5/6)

- Each data file is a **3D numpy array** of size (N, T, S) with:
  - N: total number of executions
  - T: length for 4 seconds of data (depends on the device)
  - S: number of sensor channels (depends on the device)
- The label file is a **1D numpy array** of size (N). It contains integers between 0 and 54. Each integer represents a behavioural activity.



The Cognitive Village dataset (6/6)

Label	Activity
0	Bring
1	Clean Floor
2	Clean Surface
3	Close Big Box
4	Close Door
5	Close Drawer
6	Close Lid By Rotate
7	Close Other Lid
8	Close Small Box
9	Close Tap Water
10	Drink
11	Dry Off Hands
12	Dry Off Hands By Shake
13	Eat Small
14	Gargle
15	Getting Up
16	Hang
17	Lying Down
18	Open Bag
19	Open Big Box
20	Open Door
21	Open Drawer
22	Open Lid By Rotate
23	Open Other Lid
24	Open Small Box
25	Open Tap Water
26	Plug In
27	Press By Grasp

Label	Activity
28	Press From Top
29	Press Switch
30	Put From Bottle
31	Put From Tap Water
32	Put High Position
33	Put On Floor
34	Read
35	Rotate
36	Rub Hands
37	Scoop And Put
38	Sitting Down
39	Squatting Down
40	Standing Up
41	Stand Up From Squatting
42	Take From Floor
43	Take From High Position
44	Take Off Jacket
45	Take Out
46	Talk By Telephone
47	Throw Out
48	Throw Out Water
49	Touch Smartphone Screen
50	Type
51	Unhang
52	Unplug
53	Wear Jacket
54	Write

Objectives of the project

- **Objectives**:
  - Classification of **behavioral activities** of the CogAge dataset (with both hand executions)
  - Implement the PRC to obtain a trained classifier on the CogAge dataset
  - Obtain evaluations of the performances of the classifier
- **Constraints:**
  - You must use the provided training and testing sets
  - You must use the two following evaluation metrics:
    - Accuracy
    - Average F1-score (also sometimes called F-score or F-measure)
- **Additional specifications:**
  - All classification approaches are allowed
    - If Deep Neural Networks are used, additional evaluation metric: Mean Average Precision (code provided on Moodle)
  - All Python libraries are allowed
  - The final performances of your method will **not** be taken into account for your grade!

Project organisation

- Project carried out in groups
- Each group can progress at its own pace, but reaching the following milestones is advised:
  - **1st week**: overview of the complete PRC + manipulation of the dataset + code structuring + first Python implementations
  - **2nd week**: feature extraction implementation
  - **3rd week**: classifier evaluation and obtention of performances
  - **4th week**: finalization of the implementation and writing of the report

Project evaluation (1/2)

Reminder:

- For all scenarios, the evaluation is **binary**, i.e. **pass or fail.**
- In order to pass, the following contents are expected from each group:
  - A **working** Python code of the solution
  - A report describing the implemented solution
  - A short description on how to use the code (e.g. README text file, instructions in the report, etc.)
- The deadline for the contents is **at the end of 4th week (23rd June)** for the activity monitoring scenario.

Project evaluation (2/2)

For the first scenario about activity monitoring:

- **Solution** = code which implements the PRC on the CogAge dataset and returns the performances of the trained classifier on the testing set.
- What is expected to be in the report at minima:
  - Description of the implemented PRC
  - Results obtained by the method
  - Short explanation of the structure of the Python code
- The structure and length of the report are up to you.
- **Main evaluation criteria**: scientific soundness of the proposed approach (and not the performances themselves!)

Next steps

- If you have any general questions regarding this presentation, please ask the supervisor.
- Start with 10-15 minutes discussions, then the supervisor will come to each group for a short discussion.
- If you have any question or are in need of advice, please ask the supervisor.
- The supervisor will come check your progress at regular intervals
- Do not hesitate to look online for ideas / answers to your questions.

Advised checklist for week 1

- Attribute roles within your group (e.g. who programs? checks the literature for ideas? checks the data? take care of the report? …)
- Discuss to get an overall idea of the whole PRC and how to implement it in Python
- Understand the structure of the data and how to manipulate it
- Determine which sensor(s) to use and why
- Determine what pre-processing to use to make the data suitable for further steps (if needed)
- Implementation of the pre-processing and data manipulation functions in Python

Advised checklist for week 2

- Check if an existing Python library can be used for your chosen feature extraction technique
  - If yes, check how to use it
  - If not, implement the approach by yourself
- Perform some tests to check that your feature extraction code is working properly
- Check if the features computed by your approach can be properly used as input of the chosen classifier
- Compute the features on the training and testing sets

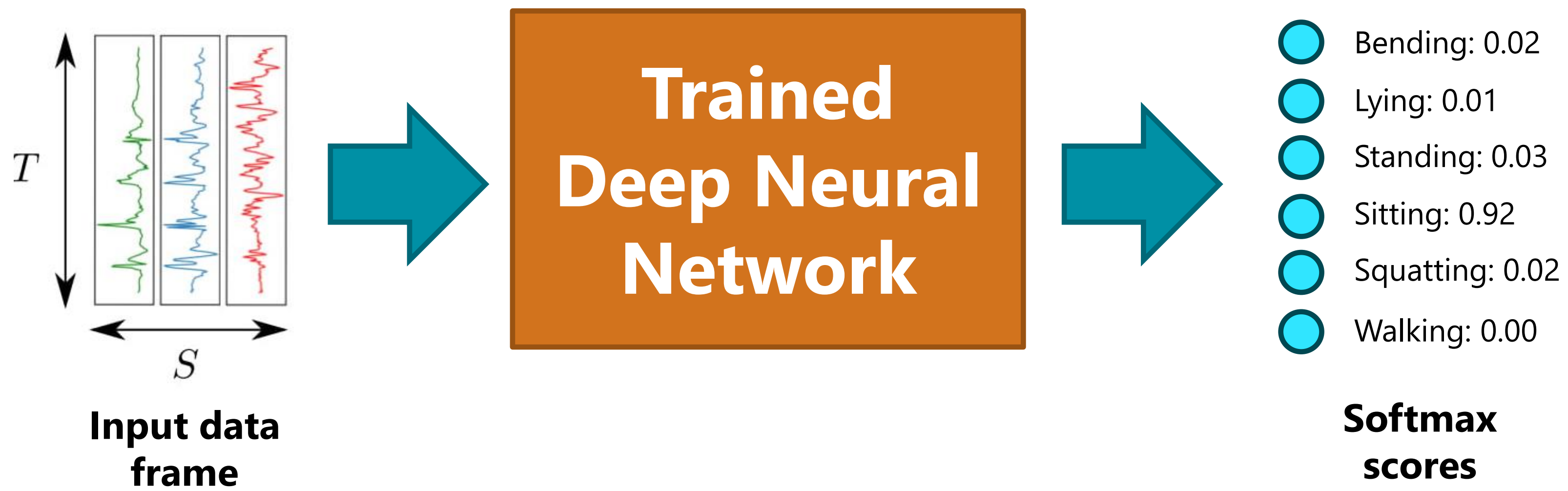
Advised checklist for week 3

- Train your classification model on the training set
  - **Note**: keep in mind a training session can take a lot of time depending on the size of the training set and your computational power
- Evaluate your trained model on the testing set using the imposed evaluation metrics (accuracy and average F1-score)
  - **Note**: do not forget to compute the Mean Average Precision as well if you used Deep Neural Networks
- Write guidelines for your Python code
- Start with the writing of the report if not already done

Annex: Mean Average Precision

Classification with Deep Neural Networks

- Deep Neural Networks (DNNs) can perform classification by using a **Softmax classification layer**:
  - N neurons in the layer, with N = number of classes
  - Each neuron outputs a value between 0 and 1
  - The sum of all neuronal outputs is equal to 1
- The output of each Softmax neuron represents the probability that the input data belongs to its associated class

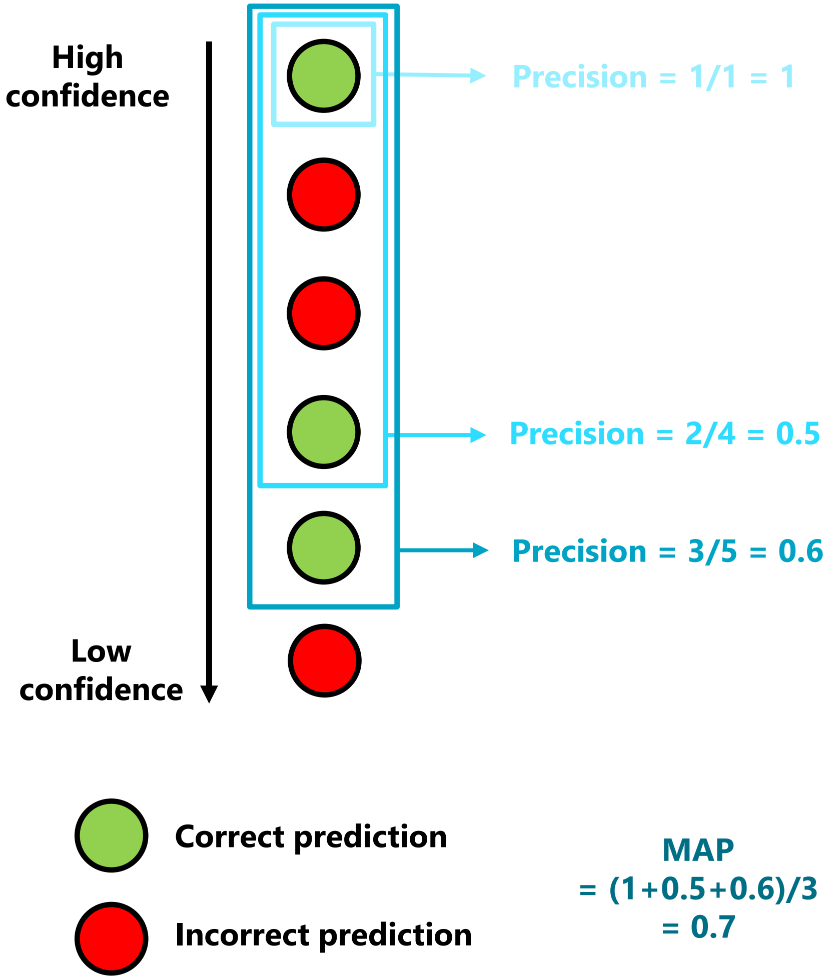


Why using Mean Average Precision?

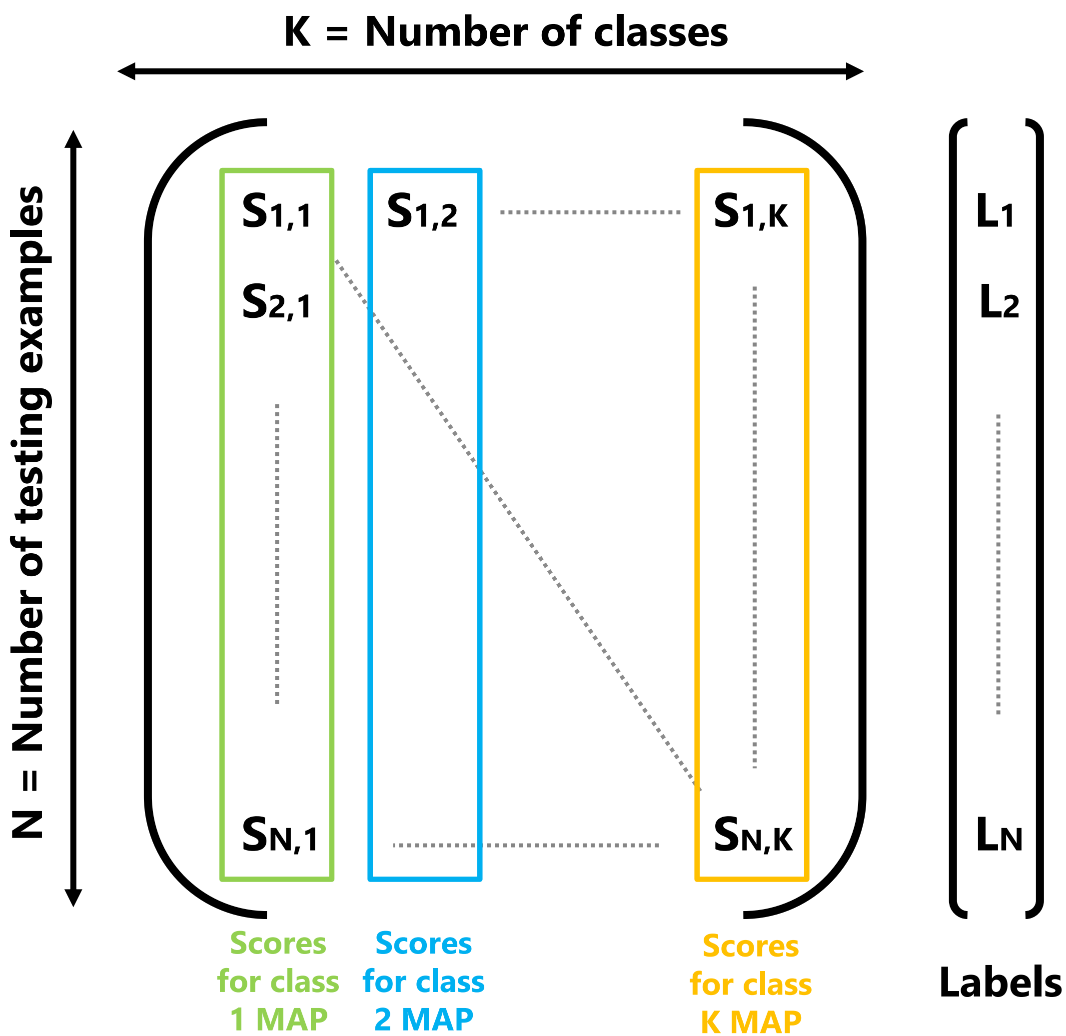
- Softmax scores can represent **how confident** a DNN is in its prediction
- It is better to obtain a DNN which is:
  - Confident when predicting correctly
  - Non-confident when predicting incorrectly
- Most standard metrics (e.g. accuracy, precision, recall, F1-score, …) cannot evaluate this
- Mean Average Precision (MAP) can be used instead in that case

Computation of Mean Average Precision

- **Step 1:** order predictions by **decreasing confidence**.
- **Step 2:** for each prediction (starting from the most confident):
  - If the prediction is incorrect: do nothing
  - If the prediction is correct: compute the **precision** using all predictions ranked higher or equal than the current one
- **Step 3:** average all precisions obtained at step 2 to get the MAP



Mean Average Precision for DNNs



- After training a DNN, it is possible to obtain class Softmax scores for all examples of the testing set
- For each class, compute a **class MAP**
- Obtain the **global MAP of the DNN** by averaging all class MAPs

Using Mean Average Precision in scenario 1

- **Note**: MAP is relevant to any approach which can return ordered classification results (DNN is only one particular example)
- Global MAP should be computed on the CogAge dataset
- No standard existing Python implementation for MAP → a Python implementation is provided on Moodle