Lecture 1. Course Introduction and Practicalities

IL2233 Embedded Intelligence

Prof. Zhonghai Lu

KTH Royal Institute of Technology

March 19, 2024

Outline

1. What is the course about?
2. Course design principles
3. Course structure, content and organization
4. Course examination
5. Practicalities
6. Demo: Sine wave prediction

What is the course about?

- This is a new course (from 2022), developed to meet the educational needs of developing and deploying various intelligent data analysis functions in sensor-based embedded systems, Internet of Things (IoT), and Cyber-Physical Systems (CPS).
  - IL2230 Hardware Architectures for Deep Learning from 2019.
- The course aims to provide the students with **essential theoretical methods and practical skills**, which are needed to develop, assess, and deploy intelligent functionalities in smart electronics and embedded systems. In particular, **dependability** and **sustainability** are considered in the course.

What to focus on?

- The course covers *selected statistical learning/machine learning/deep learning methods* for realizing relevant functionalities (e.g. feature exaction, time-series forecasting, clustering, and anomaly detection, etc.) desired in embedded systems.
- The course focuses more on *applied time-series analysis techniques*, touching upon both statistical and deep learning approaches, and *their application in embedded systems*.
  - Intelligent embedded systems often deal with time-series data from various sensors such as temperature/humidity, pressure, camera (image sensor), IMU (Inertial Measurement unit), lidar, radar, EEG (Electroencephalogram), ECG (Electrocardiogram) etc., much often than time-independent data.
  - Time-series analysis has its own set of theory and tools, in particular, from the statistical learning domain.

Course design principles

- Balance between theory and practice, but more on practice and practical skills.
- In terms of breadth and depth, the course is broader, covering a wide range of knowledge.
  In some areas, the depth is limited, leaving space for own exploration.
- Focus on understanding of basic concepts and principles rather than complicated application scenarios, so as to establish a good foundation for further investigations on your own.
- Hands-on coding examples.

Course structure

The course is designed to have 16 teaching & learning activities.

- 10 Lectures organized as 3 modules
- 3 Labs (Lab1, Lab2, Lab3 + Lab0 individual preparation for tool installation)
- 2 Seminars
- 1 Project
  A common application of time-series analysis for embedded system: Anomaly detection

Course content

- The course content is very carefully selected, balanced, and designed.
  - It has gone through many times of revision.
  - In aware of workload, relevance, significance, etc.
- The Canvas course room `https://canvas.kth.se/courses/46239`
- Let's go to the canvas course room to see
  - What are the modules, lectures?
  - What are the labs?
  - What are the seminars?
  - What is the project?

Activity organization

- 3 Labs
  - Group work, 2 students per group.
  - Lab completion: 1 technical report per group. Results check per group
  - Grouping: Automatic/Random grouping in Canvas
- 2 Seminars
  - Group work, 3-4 students per group.
  - Seminar: Being **presenter** and **opponent**
    - Each group makes presentation slides with everyone's contribution.
    - During presentation, each one presents one part of the presentation.
    - In Q & A, each group also acts as opponent group to another group.
  - Grouping: Automatic/Random grouping in Canvas
- 1 Project
  - **Individual work**, Individual completion.
  - There are two Project sessions for you to do the project tasks. Similar to Lab sessions, TAs are available for assistance.

Examination

- No written examination but the **Final Project Workshop** works as Oral Examination.
  - It is compulsory, requiring physical presence.
  - In the Project Workshop, you make slides, orally present your project individually, and answer questions.
- Three examination moments
  - Lab
  - Seminar
  - Project
- You need to pass all the 3 moments in order to pass the course
- Course grading: Pass, Fail
  - Allowing self-exploration without worrying about grades

Background survey

- This link is for students to answer the survey.
  `https://forms.gle/YmwoT59Ry7QRKjmv5`

Programming environment

- Programming language: Python, C/C++
- Basic Python libraries
  - numpy, scipy
  - matplotlib
    seaborn: Statistical data visualization `https://seaborn.pydata.org/`
  - pandas
- Statistical/Machine learning toolbox
  - Statistical learning library: statsmodels, pmdarima
  - Machine learning: scikit-learn
  - Deep learning library: keras, tensorflow

Python: A high-level programming language

- Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary.
- Python allows you to split your program into modules that can be reused in other Python programs.

Python: A high-level programming language

- Python is simple to use, and enables programs to be written compactly and readably.
- Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:
  - the high-level data types allow you to express complex operations in a single statement;
  - statement grouping is done by indentation instead of beginning and ending brackets;
  - no variable or argument declarations are necessary.
- The Python tutorial.
  `https://docs.python.org/3/tutorial/index.html`

Packages

- Packages are a way of structuring Python's module namespace by using "dotted module names".
- For example, the module name A.B designates a submodule named B in a package named A.

Numpy

- Numpy is a core library for **Scientific computing** in Python.
- Numpy provides a high-performance multidimensional array and basic tools to compute with and manipulate arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
- Numpy quick start.
  `https://numpy.org/doc/stable/user/quickstart.html`

Scipy

- SciPy, a scientific library for Python is an open source, BSD-licensed library for mathematics, science and engineering.
- The Scipy library depends on NumPy, and provides a large number of functions that operate on numpy arrays.
- Scipy user guide: an introductory tutorial, which covers the fundamentals of SciPy and describes how to deal with its various modules.
  `https://docs.scipy.org/doc/scipy/tutorial/index.html`

Matplotlib

- Matplotlib is a Python 2D plotting library. which produces **publication quality** figures in a variety of hardcopy formats and interactive environments across platforms.
- Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and graphical user interface toolkits.
- Matplotlib
  `https://matplotlib.org/index.html`

Matplotlib

- One can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code.
- For simple plotting the **pyplot** module provides a MATLAB-like interface, particularly when combined with IPython.
- For advanced use, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.
- Pyplot tutorial: An introduction to the pyplot interface.
  `https://matplotlib.org/stable/tutorials/introductory/pyplot.html`

Pandas

- Pandas is a software library written in Python for **data manipulation** and **analysis**.
- In particular, it offers data structures and operations for manipulating numerical tables and time series.
- Pandas user guide and 10 minutes to pandas
  `https://pandas.pydata.org/pandas-docs/stable/user_guide/`

Scikit-learn (sklearn)

- A **machine-learning** library or package in Python.
- It features various classification, regression and clustering algorithms including support-vector machines, random forests, k-means etc.
  - Simple and efficient tools for predictive data analysis
  - Built on NumPy, SciPy, and matplotlib
  - Open source, commercially usable - BSD license
- Scikit-learn tutorial
  `https://scikit-learn.org/stable/tutorial/index.html`

Keras: Deep Learning Library for Theano and TensorFlow

- Keras is a **deep learning** API written in Python, running on top of the machine learning platform TensorFlow or Theano. Or, Keras is a high-level API of the tensorflow/Theano platform.
- It was developed with a focus on enabling fast experimentation from training to inference.
- Keras website. `https://keras.io/about/`
- Keras means **horn** in Greek.

Keras: Deep Learning Library for Theano and TensorFlow

- The core data structures of Keras are **layers** and **models**.
- The simplest type of model is the Sequential model, a linear stack of layers.
- For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs of layers, or write models entirely from scratch via sub-classing.

Statsmodels

- statsmodels is a Python module that provides classes and functions for the estimation of many different **statistical models**, as well as for conducting **statistical tests**, and **statistical data exploration**.
- Website
  `https://www.statsmodels.org/stable/index.html`
- Tutorial
  `https://www.statsmodels.org/stable/gettingstarted.html`

General installation guide

- Download and install **Anaconda**.
  - This gives you Python along with many built-in libraries.
- Create and activate **a virtual environment**, e.g. IL2233
  `https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/`
  - This is an additional environment to the default environment.
  - This is important for keeping an environment to use a specific version of a library.
- If a package is not present, install it using either **conda** or **pip**
- Use the virtual environment
  - Activate the virtual environment
  - De-activate the virtual environment

Installation of sklearn

- Use conda
  conda install -c anaconda scikit-learn
- Use pip
  pip install scikit-learn
  pip install -U scikit-learn
  pip install scikit-learn --upgrade
  -U or --upgrade: Upgrade all packages to the newest available version.

`https://scikit-learn.org/stable/install.html`

Installation of keras

Keras comes packaged with TensorFlow as tensorflow.keras.
To start using Keras, simply install TensorFlow: pip install tensorflow

- Use conda
  conda install -c anaconda keras
- Use pip
  pip install keras
- If you already have TensorFlow and Keras installed, they can be updated and then verified by:
  pip install -U tensorflow
  python -m pip show tensorflow

`https://keras.io/about/`

Installation of statsmodels

- Installing statsmodels
  - The easiest way to install statsmodels is to install it as part of the Anaconda distribution, a cross-platform distribution for data analysis and scientific computing. This is the recommended installation method for most users.
- Python support
  - statsmodels supports Python 3.8, 3.9, and 3.10.
- Anaconda
  - statsmodels is available through conda provided by Anaconda.
  - The latest release can be installed using:
    conda install -c conda-forge statsmodels
- PyPI (pip)
  - To obtain the latest released version of statsmodels using pip:
    pip install statsmodel

`https://www.statsmodels.org/devel/install.html`

Validate your installation

- Show if you have a package, which version
  - pip show scikit-learn
  - pip show keras
  - pip show tensorflow
  - pip show statsmodels
- List all installed Python packages
  - pip list
- Show Python/module version
  - python --version
  - pip --version
  - anaconda --version

Demo: Sine wave prediction (1)

Task: Train a neural network (NN) using a synthetic dataset from a noisy y(t)=sin(t) function, and then predict the values of the sine function y given values of t.

Implementation: Generic learning/inference steps:

1. Data generation: Generate data by a noisy sin() function: $y(t) = sin(t) + c \cdot \epsilon(t)$, where $\epsilon(t)$ is Gaussian noise with mean 0 and variance 1, and c coefficient for modulating the noise.
2. Data pre-processing: Split the data set into a training set and a test set, e.g. 80% for training and 20% for testing.

Demo: Sine wave prediction (2)

1. Model definition: Define a NN model, e.g., a MLP (multi-layer perceptron) model including its hyper-parameters (number of layers and number of neurons per layer, activation function etc.) and optimization parameters (optimizer (e.g. Adam: Adaptive moment estimation), loss function).
2. Model training: Train the NN to settle its weights and biases, optimizing towards the loss function.
3. Model inference: Use the trained model to do prediction given an argument value.
4. Model evaluation: Calculate accuracy metrics of the trained model.
5. Result visualization. Visualize results.

`http://localhost:8889/notebooks/OneDrive%20-%20KTH/IPython/IL2233VT22/Lec1_intro/sine_wave_prediction.ipynb`
(The link is only used for in-classroom demo, not accessible by students.)

Homework

- Installation of Python and the packages.
  - Use Anaconda
  - Create a virtual work environment
- Go through the Python tutorial
  `https://docs.python.org/3/tutorial/index.html`
- Go through other tutorials such as numpy, scipy, matplotlib etc.
- If you are all good, try to implement the sine wave prediction task.
- **Thinking question**: What if we treat the sine wave as a **time-series** signal, and how to predict, more precisely, forecast its next value, based on its present and previous values?