VT2024: IL2233 Lab 0
Preparation

Zhonghai Lu

March 19, 2024

1 Introduction

In the course, we are going to use a variety of tools in our lab tasks. This is a preparation lab for tool installation and programming environment setup for Python and related AI libraries, and a small warm-up task.

This is an individual preparation, not group work.

This is a self-completion lab. No report is needed.

2 Installation and test

You are advised to follow the instructions in Lecture 1 Slides to install and test the following tools or frameworks.

- Download and install the package manager Anaconde.
- Create (and later use) a virtual environment for IL2233-VT24.
- Install commonly used packages such as Pandas, numpy, scipy, matploatlib.
- Install and test machine-learning library scikit-learn.
- Install and test deep-learning framework Keras, tensorflow.
- Install and test statistic-learning package statsmodels. `https://www.statsmodels.org/stable/index.html`

3 Task

The warm-up task for the course is to do Sine-wave prediction, which is discussed in Lecture 1.

Task: Build and Train (1) an MLP (Multi-layer perceptron) model, (2) a polynomial regression model, which realizes the sine function y(t)=sin(t). Given a value t, it can predict the value y(t).

Dataset: Generate by yourself. The synthetic data should contain some noise, which you can control the level of noise in the data (using a parameter).

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.

Implementation: Generic learning/inference steps, assuming you are using a neural network (NN) to do the task.

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, e.g., Mean Squared Error (MSE), of the trained model.
5. Result visualization. Visualize results.

Question:

- Now you can compare the two models. Which one gives better prediction accuracy? Why?
- What if we change the task from prediction, actually, estimation, to forecast? This means that you can forecast the next value, given current and previous values? How to re-build your model to implement this forecasting function?