why not change the world?®

Power System Machine Learning Applications: From Physics-Informed Learning for Decision Support to Interface at the Edge for Control

Luigi Vanfretti, Tetiana Bogodorova, Sergio A. Dorado-Rojas

luigi.vanfretti@gmail.com, bogodt2@rpi.edu, sergio.dorado.rojas@gmail.com

- Setup and Installation
- Data Visualization
- Data Labeling and Preprocessing
- Development of NN-based Classifier
- Traditional Machine Learning Solutions

- Setup and Installation
- Data Visualization
- Data Labeling and Preprocessing
- Development of NN-based Classifier
- Traditional Machine Learning Solutions

Download the data and the code from:

https://github.com/ALSETLab/Tutorial_SGC_2020

Install Anaconda/Miniconda:

Download - https://www.anaconda.com/products/individual
Installation - https://docs.anaconda.com/anaconda/install/

Overview

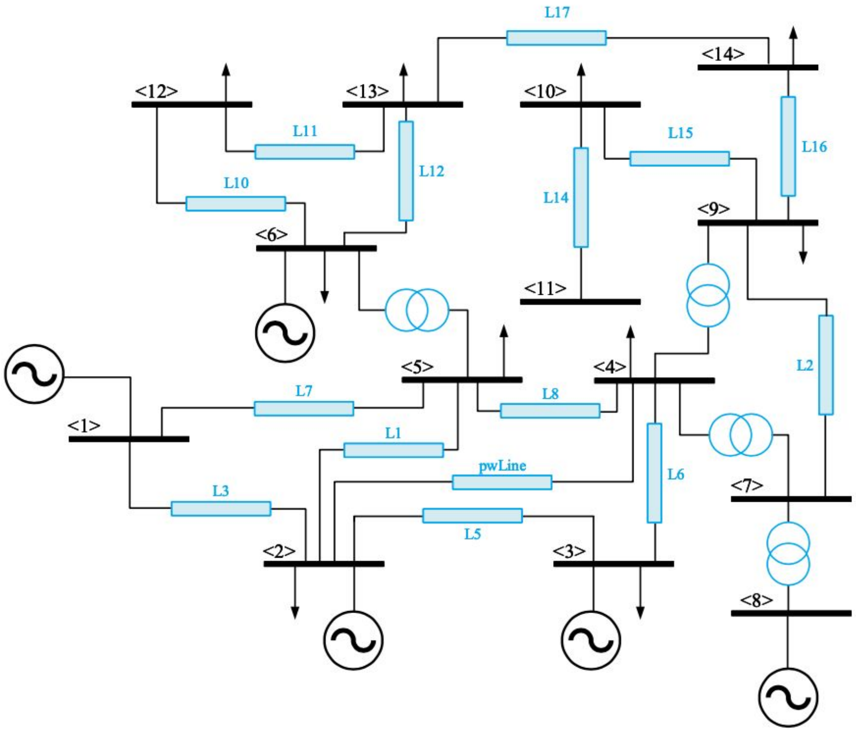
- Setup and Installation
- Data Visualization
- Data Labeling and Preprocessing
- Development of NN-based Classifier
- Traditional Machine Learning Solutions

In this part, we will:

- Load the generated data for training the ML modules

- Visualize the data and detect some problems that we need to deal with before feeding data to the ML methods

Case of Study: we use the IEEE 14 bus system for data generation



This system counts with:

- 5 generators
- 16 lines
- 4 transformers between buses

20 elements connect the system nodes

If the XFR/line impedance value of one of these element models is made large enough, we would have "applied" a contingency to the system

$$X_i \approx 10^{12}$$

By making the change of the impedance large, we do not alter the topology of the system (and do not change the number of states nor the **A** matrix)

Data Generation - Procedure

For dynamic simulation, we have employed OpenIPSL and Dymola (Modelica IDE)

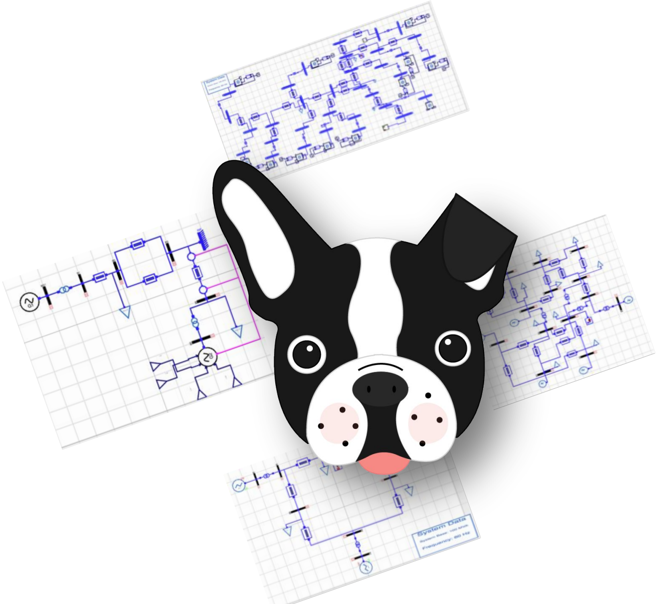
OpenIPSL is an ***open-source*** Modelica library for power systems

- It contains a set of **power system components** for **phasor time domain** modeling and simulation
- Models have been **validated** against a number of reference tools (mainly PSS/E)

Python Dymola Interface

Scenario Selection → **Model Setup** “Opening” branches → `linearizeModel` → **Scenario Eigenvalues**

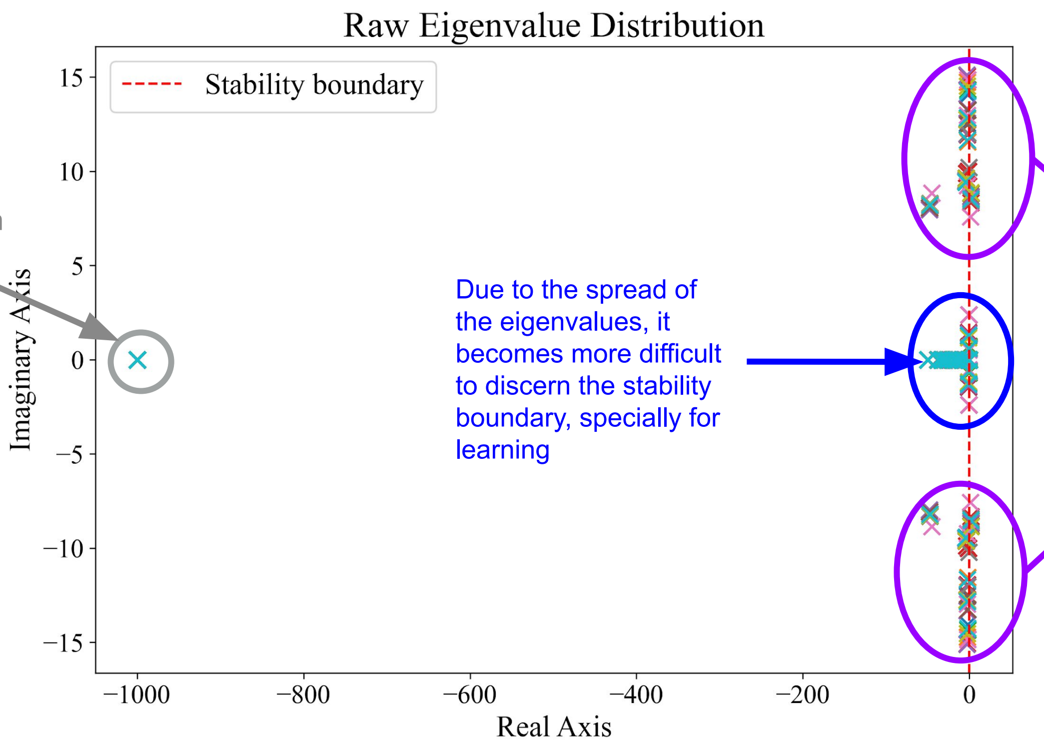
20,000 Scenarios



https://github.com/OpenIPSL/OpenIPSL

- Setup and Installation
- Data Visualization
- Data Labeling and Preprocessing
- Development of NN-based Classifier
- Traditional Machine Learning Solutions

Data Generation - Results



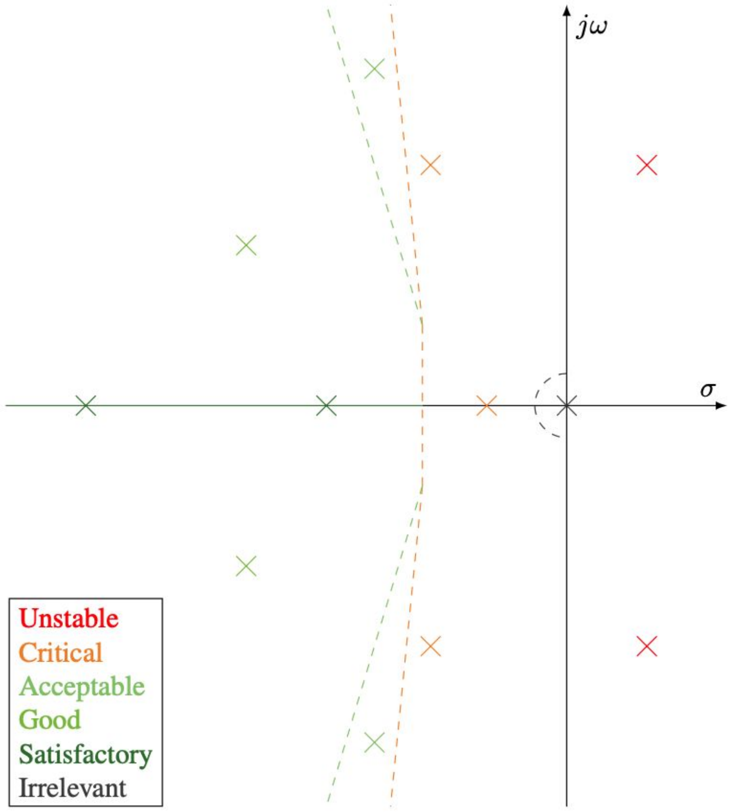
Too far away from the origin

Due to the spread of the eigenvalues, it becomes more difficult to discern the stability boundary, specially for learning

Eigenvalues too spread along the imaginary axis

Damping ratio computation can be hard-coded (*hard-coded classifier*) and the resulting vectorized function is used to *label* the eigenvalues in the following groups:

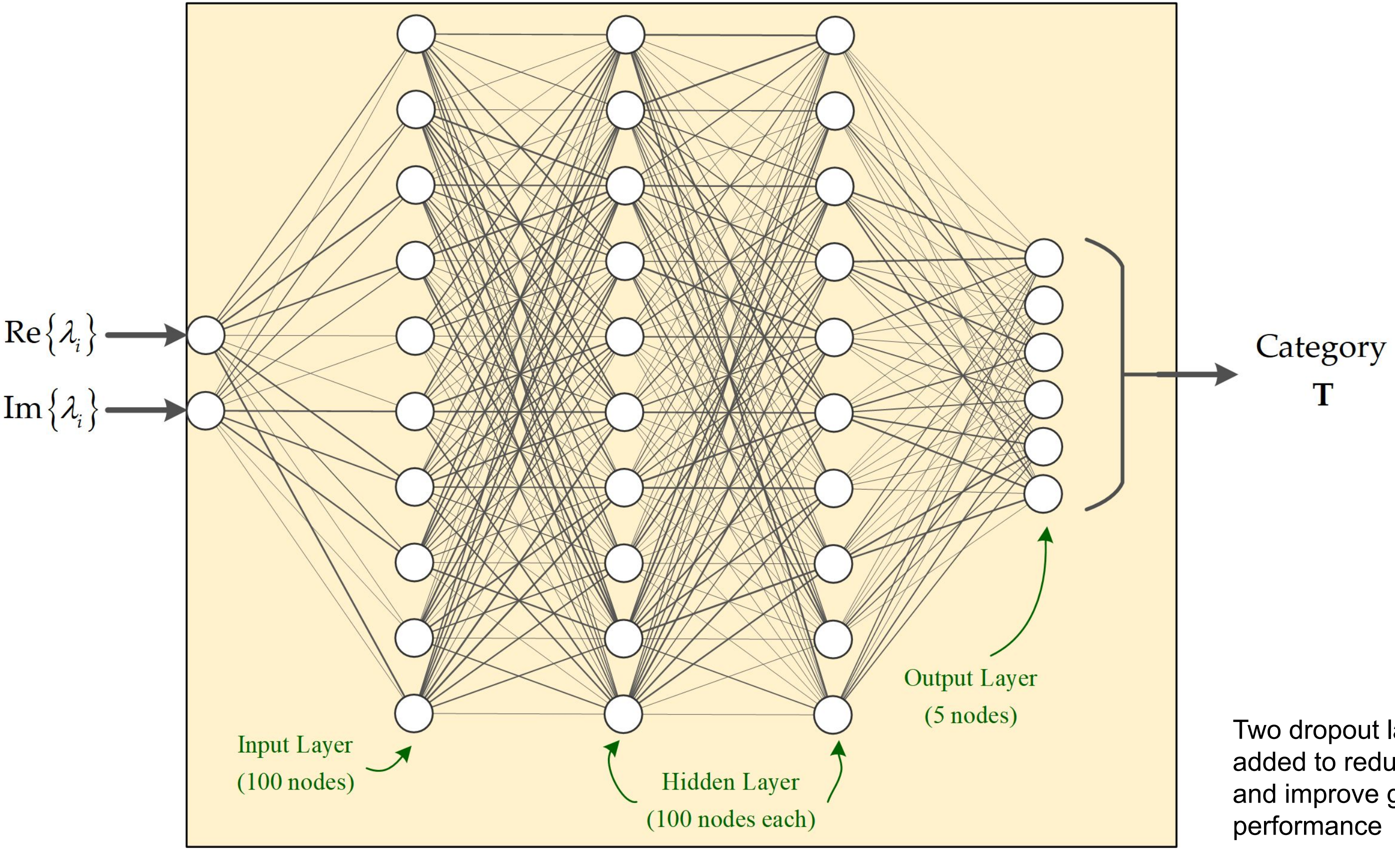
- **Unstable** $(\zeta < 0)$
- **Stable but critical** $(0 \leq \zeta < 0.05)$
- **Acceptable** $(0.05 \leq \zeta \leq 0.1)$
- **Good Operation** $(0.1 \leq \zeta < 1)$
- **Satisfactory Operation** $(\zeta \geq 1)$
- **Irrelevant**



Overview

- Setup and Installation
- Data Visualization
- Data Labeling and Preprocessing
- Development of NN-based Classifier
- Traditional Machine Learning Solutions

Neural Network Design



Two dropout layers are added to reduce overfitting and improve generalization performance

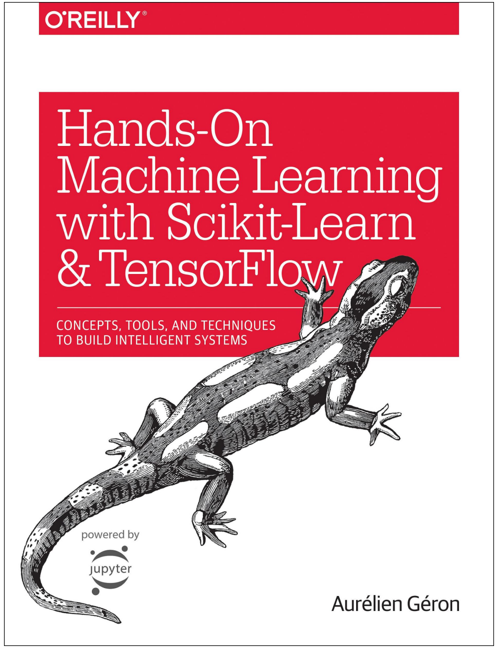
- Setup and Installation
- Data Visualization
- Data Labeling and Preprocessing
- Development of NN-based Classifier
- Traditional Machine Learning Solutions

Traditional Machine Learning techniques are trained and benchmarked against NN for eigenvalue classification

Selected techniques:

- Logistic Regression
- Softmax Regression
- Support Vector Machines
- *k*-Nearest Neighbors
- Decision Trees
- Naive Bayes

All of the algorithms were implemented and tested using `scikit-learn`



Rensselaer

why not change the world?®