APSSS

POWER SYSTEM SCHOOL

APSSS 2020 – Homework Assignment

"Machine Learning for Power System," prepared by *Andreas Venzke*, *Jochen Stiasny*, and *Dr Spyros Chatzivasileiadis*, Technical University of Denmark
"Reinforcement Learning for Demand Response," prepared by *Jose R Vazquez Canteli*, University of Texas at Austin
"Clustering and Classification for Power System," prepared by *Dr Archie Chapman*, The University of Queensland

Note: Please return your report before the end of March 6th, 2020 to theapsss@adelaide.edu.au.

February 20, 2020

About homework assignment

The homework exercises are developed by the speakers based on the materials and tools covered during their lectures. You are expected to write a report by answering the questions asked by our speakers in this document (Sections 1.3, 2.1, and 3.1). In order to get the course credit, you need to get an average of 65% on all three topics, or by choosing two topics out of the three and get at least 80% on each. For those of you who are interested to be considered for the APSSS Student Awards, the same roles apply, and the awards will go to the three top students with the highest scores. Course credit seekers will be considered for the awards automatically.

For your convenience, all the recordings are made available for Day 1, Day 2, Day 3, and Day 4. **Please do not share the links with anyone. These are uncut and unedited recordings and should not be shared with public. The lecture recordings will be publicly available later after editing.**

The report submission deadline is by the end of **March 6th, 2020** (Adelaide time). The deadline will not be extended in any circumstances. Please return your report to theapsss@adelaide.edu.au. If you have any question or concern, please contact us at theapsss@adelaide.edu.au. *Do not forget to specify whether you want the course credit or not in your email and report.*

1 Machine Learning for Power Systems

1.1 Introduction

This assignment has two goals. First, to implement a Decision Tree and a Neural Network that can assess the security of a power system, i.e. determine if any given operating point is safe or unsafe. Second, to get familiar with physics-informed neural networks.

The objectives to be achieved at the end of the assignment are the following:

- Decision Trees and Neural Networks for Power System Security Assessment
  - Get familiar with the database
  - Implement a Decision Tree
  - Implement a Neural Network
  - Investigate the impact of different properties of Neural Networks on the classification accuracy
  - Compare the accuracy of the Decision Tree vs. the accuracy of the Neural Network
- Physics-Informed Neural Networks
  - Get familiar with the code
  - Identify strengths and opportunities for improvements (aka shortcomings :)) of the physics-informed neural networks for continuous time

1.2 Preparation

Make sure you have all the required modules installed, as follows:

1. `scikit-learn`
2. `tensorflow`
3. `matplotlib`
4. `pyDOE`
5. Examples of commands:
   - if you run Python on Mac: `pip install PackageName` in a terminal
   - if you are on Windows with Anaconda, then `conda install PackageName` in Anaconda Prompt with "Run as an Administrator" (not the Windows command prompt)

1.3 Tasks

Note: All files and code can be downloaded in a single file from http://www.chatziva.com/downloads.html

1.3.1 Decision Trees (DT) and Neural Networks (NN) for Power System Security Assessment

1. Download the training database. Database prepared by Florian Thams, Andreas Venzke, and Lejla Halilbasic. See the end of the document for more information about the database in Section 1.4.
2. Download the python .py file `DT_and_NN_Security_Assessment_14bus.py`
3. Go through the code, try to understand it, and add a comment to every command to explain its function (you can also run it, this should help your understanding).
4. Train a decision tree and a neural network and evaluate their accuracy.
5. Is the database balanced? Is accuracy a sufficient metric? If not, evaluate the different performance metrics, e.g. Matthews correlation coefficient.

6. Experiment with different neural network topologies and activation functions (e.g. ReLU, sigmoid, etc); Report your findings on how this impacts the accuracy metrics.

7. Compare the performance metrics for the decision tree and the neural network you trained. Which has the best performance? What are the advantages and disadvantages of each method?

Optional Tasks:

8. Experiment with the different datasets, after you have trained your first decision tree and neural network for only N-1 security and for both N-1 security and small signal stability.

9. The decision tree and the neural network are bound to have some classification errors. Implement an N-1 AC security assessment to validate parts of the database and check if there are any misclassifications.

1.3.2 Physics-Informed Neural Networks (PINN)

1. Download the python .py file `PINN_inference_swing_equation.py`

2. Go through the code, try to understand it, and add a comment to every command to explain its function (you can also run it, this should help your understanding). The following references might (or might not) be helpful:

   - G. S. Misyris, A. Venzke, S. Chatzivasileiadis, Physics-Informed Neural Networks for Power Systems. 2019. https://arxiv.org/pdf/1911.03737.pdf
   - M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-Informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations", Journal of Computational Physics, vol.378, pp. 686-707, 2019 https://arxiv.org/abs/1711.10561

3. Set the number of collocation points to Nf=100. How is the PINN performance and the computation time?

4. Set the number of collocation points to Nf=1000. How it the PINN performance compared to the previous task?

Optional Task:

5. If you have a powerful laptop, set the number of collocation points to Nf=10000 (this will take some time, maybe 1 hour). How it the PINN performance now?

1.3.3 General Questions

1. The DT and NN in Section 1.3.1 perform a different function from the NN in Section 1.3.2. What is their main difference? (hint: think about the outputs)

2. What was the size of the external training data you needed for the training of the Decision Trees and the Neural Networks in Section 1.3.1 and what was the size of these external data in Section 1.3.2? In general, would you expect that a classification or a regression neural network would require more training data?

3. What are your conclusions about the impact of a balanced training database on the DT and NN training?

4. What are the benefits and the shortcomings of Physics-Informed Neural Networks? Where should future research focus, in order to remove the barriers for real-life application of these methods?

1.4 Database

You will receive two datasets that were prepared for the IEEE 14-bus system. You are welcome to use these datasets for your future studies, your work or your research. If you use them please cite the following paper:

F. Thams, A. Venzke, R. Eriksson and S. Chatzivasileiadis, "Efficient Database Generation for Data-Driven Security Assessment of Power Systems," in *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 30-41, Jan. 2020. doi: https://arxiv.org/pdf/1806.01074

L. Halilbasic, F. Thams, A. Venzke, S. Chatzivasileiadis, P. Pinson. "Data-driven Security-Constrained AC-OPF for Operations and Markets,"In 20th Power Systems Computation Conference, Dublin, Ireland, pages 1-7, June 2018. https://www.researchgate.net/publication/327331498_Data-driven_Security-Constrained_AC-OPF_for_Operations_and_Markets

1.4.1 Difference between the two datasets

Database OPF with VG: Q-limits are not enforced. For this dataset, we run the standard power ow algorithm (provided by Matpower), which does not check if the PV buses violate their reactive power limits.

Database OPF without VG: Q-limits are enforced. For this dataset, we run a power ow algorithm (again provided by Matpower), which checks if the PV buses violate their reactive power limits. If a PV bus (which is usually a generator bus) injects reactive power that exceeds the Q-limits of the generator, the PV-bus is transformed to a PQ-bus, with Q = Qlimit, and the voltage is allowed to vary. This is a more realistic implementation of the power flow, as in reality, if the determined reactive power cannot be provided, then the voltage will necessarily change.

1.4.2 Contents of each database

Database OPF with VG: 49615 points, Q-limits not enforced

Database OPF without VG: 675367 points, Q-limits enforced
For every database, we assessed each operating point for both N-1 security and small signal stability. For more information on the method, please see F. Thams, A. Venzke, R. Eriksson and S. Chatzivasileiadis, "Efficient Database Generation for Data-Driven Security Assessment of Power Systems," in *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 30-41, Jan. 2020. doi: https://arxiv.org/pdf/1806.01074

N-1 security: We run a power flow considering the base case and the single outage of each component. If any of the power flow cases violate component limits (line flow limits or voltage limits) the setpoint is classified as N-1 insecure. Considered contingencies include all line outages (except for lines 7–8 and 6–13 that make the problem infeasible, i.e. 14–bus system is not N-1 secure for these outages).

Small-signal stability: We consider a full dynamic model for each generator (6th-order), including governor, Automatic Voltage Regulator (AVR type I, 3-states), and Power System Stabilizer (PSS). We set the stability limit at 3% damping ratio. All operating points with a damping ratio below 3% are considered insecure. For more info and the data assumed, please see F. Thams, A. Venzke, R. Eriksson and S. Chatzivasileiadis, "Efficient Database Generation for Data-Driven Security Assessment of Power Systems," in *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 30-41, Jan. 2020. doi: https://arxiv.org/pdf/1806.01074

2 Reinforcement Learning for Demand Response

Using `CityLearn` in central agent mode (just like we did in class), control building 3 under the climate zones 1, 2, 3, and 4 to minimise the following metrics: ramping, 1-load_factor, the average daily peak, the peak demand, and the net electricity consumption. Use either `SAC` or `PPO2`, from the `Open AI stable baselines`. You can tune the algorithm differently for each of the different 4 climate zones. Feel free to select any states you want (you can do this in the file `buildings_state_action_space.json`). Remember you can adjust the number of episodes by modifying the parameter `total_timesteps in the model.learn()` function.

2.1 Tasks

1. Tune the hyperparameters (i.e. `gamma`, `learning rate`, `batch_size`, `tau`) of the chosen algorithm until it is able to learn and achieve good performance. You can use the same reward function that comes with the environment (which should increase as the electricity consumption becomes flatter).

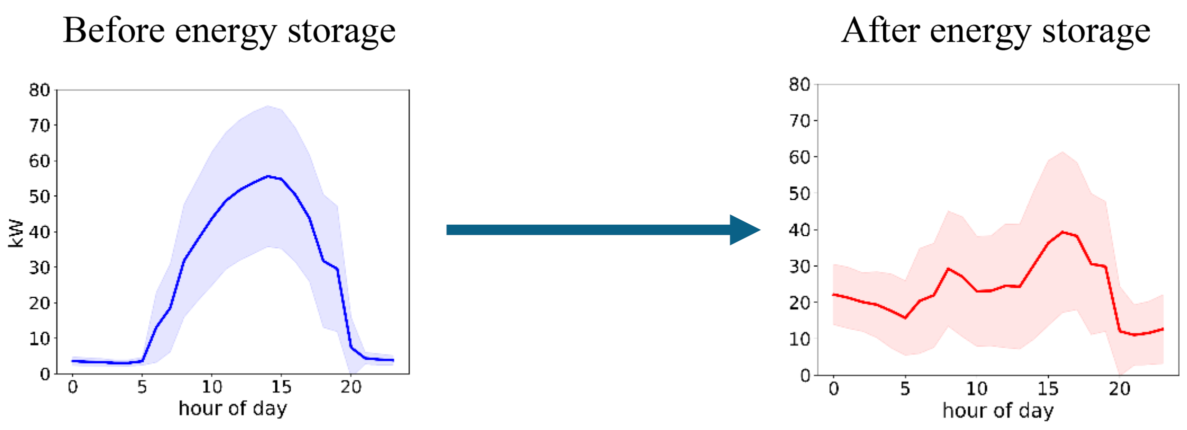
   a. Provide the reward and the performance metrics obtained on the best run of the algorithm for each climate zone.

   b. How do the different hyperparameters affect performance?

   c. Which parameter(s) increase the computing time required by the algorithm? Why?

   d. What set of states did you choose? Why?

2. Plot the "average day" profile before and after using energy storage with RL. You should find the average net electricity consumption for every hour of the day and its standard deviation. Every hour in this plot should be the result of averaging 365 values of that specific hour for every day of the year. You may use the variables `env.net_electric_consumption` and `env.net_electric_consumption_no_storage`. You can use the results of the best run of your RL controller to make each plot. Make one plot for every climate zone. Here is an example of how one of the 4 plots should look like:



3. How do the results differ between different climate zones? Do similar hyperparameters work well for different climate zones or do they need to be specifically tuned for each climate zone?

4. Do all the 5 performance metrics reach similar values as the rewards increase (as the rewards become less negative)? Why/why not? Remember, when a performance metric is lower than 1, it means that it is better than the one obtained by the rule-based controller.

For reference and documentation please visit https://github.com/intelligent-environments-lab/CityLearn

3 Clustering and Classification in Power System

First, ensure that you have "APSSS_examples_Chapman_ClusteringClassification.zip" file unzipped on your machine. The code contains two new methods, namely `generate_noisy_data.py` and `generate_corr_noisy_data.py`.

As indicated by the filenames, these two methods add noise to the "clean" voltage data set. Here are more details about the new codes:

- The first file, i.e., `generate_noisy_data.py`, adds independent and identically distributed (i.i.d.) standard Gaussian noise (zero mean, unit variance, in volts).
- The second file, i.e., `generate_corr_noisy_data.py`, add exponentially-correlated standard Gaussian noise (zero mean, unit variance, decay constant of 2.0, in volts). This statistically links the data across the column index, which matches with the geographic topology of the underlying feeder – the customers are ordered by distance from the feeder head.
- In both cases, the individual perturbations to the clean data – their marginal distributions – follow a Gaussian distribution (and if you don't believe it, generate some noise using each and compare their histograms).
- For both noise generator methods, there is a `scale` parameter, which proportionally adjusts the magnitude of the noise, i.e. scale=2 results in noise with variance 2V.

3.1 Tasks

The assignment is as follows:

1. Generate a new noisy data set with i.i.d. noise.
   a. Cluster the new data, see how this differs from the example given in the class.
   b. Using the label data in `phase_ID.csv`, build a classifier for the data. You can use the multinomial logistic classifier provided or another method. Compare the loss function value to the case with clean data. Also, compare the target accuracy. Comment on your findings.
   c. Increase the scale parameter on the noise generator, reassess the loss and target accuracy. With reference to the law of large numbers and/or the central limit theorem for i.i.d. random variables, can you explain the classifier's performance?
   d. Find a noise level where both begin to perform poorly. Is this a reasonable representation of reality? Would you expect to see this level of noise in real-world data?
2. Generate a new noisy data set with exponentially correlated noise.
   a. Go through steps 1.a to 1.d again, for the correlated noisy data.
   b. What differences to the i.i.d. noise case do you observe? Explain why the difference is occurring.
   c. Given you know the correlations are related to geographic spread, name some real-world effects that could be driving these differences.
   d. How might you incorporate these correlations into your classifier, so that you can remove them and identify the underlying phase relationships? There is no correct answer, and I can think of several approaches off the top of my head, so get creative.