Assignment: Homework 3

How to Hand It In

1. Put all your solutions in one folder. Compress this folder and name it <firstname>_<lastname>_HW3.zip. For example, "Zhiyao_Duan_HW3.zip".
2. Submit to the corresponding entry on Blackboard.

When to Hand It In

It is due at 11:59 PM on the date specified on the course calendar. **Late assignments will receive a 20% deduction of the full grade each day.**

Problems (10 points in total)

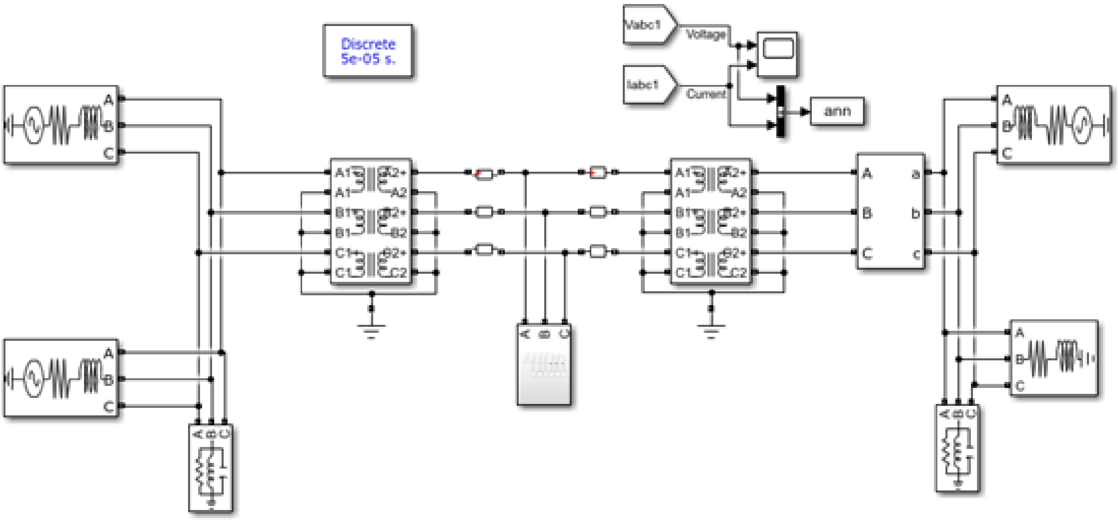
1. (1 point) Load two datasets with `sklearn.datasets.make_blobs` and `sklearn.datasets.make_moons`, each with 100 samples for positive and negative classes respectively. Visualize them. Are they linearly separable?

2. (3 points) Perceptron is a simple linear classifier described as
$$y = sgn(w^T x),$$
where x is the expanded feature vector, w is the weight vector, and y is the binary output of perceptron. The training of perceptron has different ways.
    a. (1.5 points) Implement perceptron with the "perceptron learning rule", the single-sample stochastic gradient descent update rule. Run this algorithm on both datasets. Does your algorithm converge to a good solution? Draw the classification boundary on the data visualization graph. If not, explain why not.
    b. (1.5 points) Implement perceptron with the "delta rule", the all-sample gradient descent update rule. Run this algorithm on both datasets. Does your algorithm converge to a good solution? Draw the classification boundary on the data visualization graph. If not, explain why not.

3. (3 points) Logistic Regression
    a. (1.5 points) Implement logistic regression with the "squared error loss" using gradient descent. Run this algorithm on both datasets. Does your algorithm converge to a good solution? If so, draw the classification boundary on the data visualization graph. If not, explain why not.
    b. (1.5 points) Implement logistic regression with the "log error loss" using gradient descent. If interested, you can also try the Newton's method. In that case, you would need to compute the Hessian matrix and perform matrix inverse. Run this algorithm on both datasets. Does your algorithm converge to a good solution? Draw the classification boundary on the data visualization graph. If not, explain why not.

4. (3 points) Predict faults of a power transmission line. In this part, instead of implementing your own, you will use existing classifiers provided in sklearn. The file `powerline_dataset.csv` contains several simulations of the circuit in the figure under

normal and faulty circumstances (the circuit is provided for illustration only; Knowledge of electrical circuits or transmission lines is NOT required for this assignment). Columns `Va`, `Vb`, and `Vc` contain the voltages of phases A, B, and C. Columns `Ia`, `Ib`, and `Ic` contain the respective current. Column `Output` contains 1 for faulty conditions, 0 otherwise.



a. (0.5 point) Load the dataset and divide the dataset into training, validation and testing sets.

   You may use `sklearn.model_selection.StratifiedShuffleSplit`.

b. (0.5 point) Train an SVM classifier to predict faulty vs. normal conditions based on the input voltages and currents. Tune SVM hyperparameters on the validation set. After you have finished tuning, report the classification accuracy on the training, validation and test sets.

c. (0.5 point) Train a Decision Tree to perform the same task. Tune the hyper-parameters on the validation set. After you have finished tuning, report the classification accuracy on the training, validation and test sets.

d. (0.5 point) Train a Logistic Regression classifier to perform the same task. Tune the hyper-parameters on the validation set. After you have finished tuning, report the classification accuracy on the training, validation and test sets.

e. (0.5 point) Feature Engineering. In this part, you will try to improve the Logistic Regression model with feature engineering. See skeleton notebook for examples and instructions.

f. (0.5 point) Compare the results of all models you implemented. Try to explain the performance difference. Compare the pros and cons of these models/approaches involved.

NOTE: Submit your code as Jupyter Notebook AND (optionally, for failback) as a PDF (you can export as PDF/**Print** from modern browsers). Your code should be well commented and easy to read. Make sure to include any special instructions to run your code, e.g., additional files, folders, libraries, etc.