ECE 805 – MACHINE LEARNING
Spring 2021
Homework no. 2
(due on Monday, 26 April 2021 by 1:00pm, submitted via Teams)

General instructions

This assignment consists of two questions. You are requested to submit both your code and a report; the report should contain the requested visualisation plots and your answers to possible questions.

1 (50%) Online Gradient Descent (OGD)

For this exercise you will use the *Circle* dataset from Tutorial 4, and a neural network. In Tutorial 4, we have also defined the Classifier class in Python. We recommend to use that, but you can use MATLAB, or any other programming language you wish to. The optimisation algorithm you will use is Online Gradient Descent (OGD), where its Python code have also been provided. Use the following configurations:

circle_params = {*'dataset_name': 'circle', 'num_features': 2*}
nn_params = {*'cls_name': 'neural_network', 'momentum': 0.9, 'hidden_units': 8*}

1. Use the neural network configuration described earlier, fix the activation function in the hidden layer to 'tanh', set the learning rate to 0.1 and run the simulation experiment once. Repeat with learning rates of 0.01 and 0.001.
   (a) Present a **single** figure / plot with three learning curves i.e. one for each simulation experiment. The x-axis should display the time steps and the y-axis the prequential accuracy. Each plot should look like an upward-right curve.
   (b) Describe your findings.
2. Use the neural network configuration described earlier, fix the learning rate to 0.01, and experiment with these activation functions in the hidden layer: 'tanh', 'relu', and 'sigmoid'.
   (a) Present a **single** figure / plot with three learning curves i.e. one for each simulation experiment, like in the previous question.
   (b) Describe your findings.
3. Define one-pass learning. What are its advantages and disadvantages?

2 (50%) OGD with active learning

In this exercise you are requested to incorporate active learning (AL), where it is no longer assumed that the class label becomes available after each prediction. Specifically, the classifier chooses a specific example as it arrives according to an AL strategy, and requests its class label from a human expert.

1. You will implement one of the simplest AL strategies, where **the label of an arriving instance x^t at time t, is requested with a probability** p. Your code should look like the one displayed below.

   Note: Fix the random state/seed first as follows: *random_state = np.random.RandomState(seed=0)*

Algorithm 1 OGD with active learning

h^0: neural network
for each time step $t \in [1, \infty)$ **do**
    receive example x^t
    predict class label $\hat{y}^t = h^{t-1}(x^t)$
    $h^t = h^{t-1}$
    if AL strategy requests the label of x^t **then**
        receive class label y^t
        calculate loss $L = l(y^t, \hat{y}^t)$
        update classifier $h^t = h^{t-1}.train(L)$

2. Use the neural network configuration below and experiment with these probabilities: $p = 1.0, 0.5, 0.1$.
   nn_params = {'cls_name': 'neural_network', 'learning_rate': 0.01, 'momentum': 0.9, 'hidden_units': 8, 'hidden_activation_fun': 'tanh'}

   (a) Present a **single** figure / plot with three learning curves i.e. one for each simulation experiment, like in the previous questions.

   (b) Describe your findings.

3. So far we haven't considered nonstationarity or concept drift. What are three categories of methods to address this problem?