Question 2. IWAL algorithm implementation (50 points)

The purpose of this question is to implement Importance Weighted Active Learning (IWAL) algorithm. For this question, you will not use modAL, but instead will implement IWAL routine from scratch using scikit-learn, NumPy and native Python. In this question, we will use a simple synthetic dataset for a binary classification problem. Each data point has only 2 features. The dataset is provided in 2 files -- "data_iwal.npy", which contains features and "labels_iwal.npy", which contains labels. For simplicity, you will implement bootstrapping rejection sampling subroutine with logistic regression and hinge loss.

Complete the code under ###TO DO in each cell and produce the required plots. Feel free to define any helper functions as you see fit. You may import and use any modules in scikit-learn and NumPy to help with your implementations.

Imports

Here we import necessary modules. Feel free to add something else here if you need it!

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import hinge_loss, log_loss
from sklearn.linear_model import LogisticRegression
```

Reading data

Here we read the data and split it into train and test datasets. Train will be used to train our classification model and test will be used to validate the performance, monitor overfitting and compare the results of the model trained with Active Learning with the ones of the model trained from scratch. We set aside 1/3 of the dataset for validation.

In [ ]:
```
X = np.load("HW2/data/q2/data_iwal.npy")
y = np.load("HW2/data/q2/labels_iwal.npy")
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Part 2.1

Type your answers for the theoretical questions below.

1. What is the idea behind IWAL algorithm?

Your answer goes here

1. What are the assumptions made for the IWAL algorithm?

Your answer goes here

1. What are the pros and cons of IWAL algorithm?

Your answer goes here

Part 2.2 Implement IWAL algorithm

In this part you will implement a functiom that performs a single query of Algorithm 1 IWAL (subroutine rejection-sampling) from the paper. Below is the function description that you can follow in your implementation.

In [ ]:

```
### TODO: Implement Algorithm 1 from the paper
def IWAL_query(x_t, rejection_sampling, history, h, **kwargs):
    """
    This function implements a single query if IWAL algorithm from the paper
    by Beygelzimer et al https://arxiv.org/pdf/0812.4952.pdf
    Args:
        x_t: currently considered sample
        y_t: label for x, will be added to history if requested
        rejection_sampling: python function, that accepts current data point x_t and
        some arbitrary arguments and returns probability of requesting a label for x_t
        history: dictionary of previous history. history.keys() will return dict_keys(['X',
        where, X -- is a history of sampled data points, y -- labels for samples in X, c --
        samples in X.
        h: scikit-learn model class, such as sklearn.linear_model.LogisticRegression, etc
        **kwargs: dictionary of arbitrary arguments that may be required by rejection_sampl
    Returns:
        h_t: object of scikit-learn model class h, that is optimal at current time step.
    """
    ### Your code goes here
    # p_t = ...
    # Q_t = ...
    # if Q_t = 1:
    #    ...
    # else:
    #    ...
    # h_t = ...
    return h_t
```

Part 2.3 Implement bootstraing rejection sampling subroutine

In this part you will implement bootstrapping rejection sampling subroutine from the paper, section 7.2

In [ ]:

```
def bootstrap(x, H, labels, loss, p_min=0.1):
    """
    This function implements bootstrap rejection sampling subroutine.
    Args:
        x: array-like object of features for currently considered sample
        H: list of hypothesis (scikit-learn objects) that are used in voting
        labels: list of possible labels for the problem. If binary classifiction, labels=[0
        p_min: minimum threshold for the probability
    Returns:
        p_t: probability of requesting the label for x, which is equal to:
            p_min + (1 - p_min)(max_{y, h_i, h_j} L(h_i(x), y) - L(h_j(x), y))
    """
    max_value = -10000
    for i in range(len(H)):
        for j in range(len(H)):
            for y in labels:
                # TODO: calculate max_value = max_{y, h_i, h_j} L(h_i(x), y) - L(h_j(x), y)
    return p_min + (1 - p_min)*max_value
```

Part 2.4 Organize all implemented parts into a single pipeline

Now you implemented all part of IWAL algorithm with bootstrap rejection sampling and can organize it into a pipeline

In [ ]:
```
history = {}
losses = []
n_initial = 10
n_h = 10

X_training, y_training = X_train[:n_initial], y_train[:n_initial]

#Initialization: initialize history and H
## TODO: your code here

# Create n_h classifiers and train them on bootstrapped data (data is sampled with replacem
## TODO: your code goes here

# Perform queries and record loss
n_query = 50
for t in n_query:
    ### TODO: your code goes here
    log_loss(y_test, h_t.predict_proba(X_test))
```

In [ ]:
```
plt.plot(losses)
```

Part 2.5 Compare results of Active Learning vs No Active Learning

In this part you need to create object of the same scikit learning class and train it on randomly selected subset of data points and compare results of 2 classifiers. Comment on your observations

In [ ]:
```
# Compare to no Active Learning setting

## TODO: your code goes here
```

In [ ]:
```

```