Question 1. ZLG algorithm implementation (50 points)

You are to implement the ZLG algorithem for this problem. We will use a subset of multiclass data where the label is a protein subcellular localization. The 8 features are extracted from the protein sequence. For this problem we are only using points with labels 'MIT' or 'NUC'.

First, read the paper and answer the following questions.

1. What is the idea behind the ZLG algorithm (5 points)?

2. What are the assumptions behind the ZLG algorithm (5 points)?

3. What are the pros and cons of the ZLG algorithm (5points)?

In [ ]:
```python
import numpy as np
import pandas as pd
from scipy.spatial import distance_matrix
from sklearn.preprocessing import LabelEncoder
```

A total of 892 data points have labels 'MIT' (244) or 'NUC' (429). We start with the labels of only the first 200 data points (set Y_k). The other 792 points are in Y_u.

In [ ]:
```python
data = pd.read_csv('data/data.csv')
data_CYTNUC = data.loc[data['Label'].isin(['MIT','NUC'])].values
X = data_CYTNUC[:,:8]
y = LabelEncoder().fit_transform(data_CYTNUC[:,-1])

n_l = 200

Xk = X[:n_l,:]
Yk = y[:n_l]
Xu = X[n_l:,:]
Yu = y[n_l:]
```

(5 points) Let's first construct the weight matrix W. Use t = 0, σ as the standard deviation of X. Then calculate the D matrix and the Laplacian matrix (Delta).

In [ ]:
```python
def Laplacian_matrix(X):
    ## TODO ##

    return Delta

Delta = Laplacian_matrix(X)
```

(5 points) Now complete the subroutine to compute the minimum-energy solution for the unlabeled instances. (Hint: Use the formula in page 38, Lecture 7.) The function also outputs one submatrix that we will use to select points to query.

In [ ]:
```python
def minimum_energy_solution(Delta,n_l,fl):
    """
    Args:
        Delta: The Laplacian matrix.
        n_l: Number of labeled points. Notice that Delta should have the upper left submatrix
            corresponding to these n_l points. So when new points get labeled, you may need
            to rearrange the matrix.
        fl: Known labels.
    Returns:
        Delta_uu_inv: Inverse matrix of the submatrix corresponding to unlabeled points.
        fu: Minimum energy solution of all unlabeled points.
    """
    ## TODO ##

    return Delta_uu_inv, fu

Delta_uu_inv, fu = minimum_energy_solution(Delta,n_l,Yk)
```

(15 points) We would like to query the points that minimize the expected risk. To do so, we want to be able to calculate the expected estimated risk after querying any point k. The variable Rhat_fplus_xk refers to $\hat{R}(f^{+x_k})$. fu_xk0 is $f_u^{+(x_k,0)}$ and vice versa for fu_xk1.

In [ ]:
```python
def expected_estimated_risk(Delta_uu_inv,k,fu):
    """
    Args:
        Delta_uu_inv: Inverse matrix of the submatrix corresponding to unlabeled points.
        k: index of one unlabeled point with respect to the uu submatrix (not the entire Delta)
        fu: Minimum energy solution of all unlabeled points.
    Returns:
        Rhat_fplus_xk: Expected estimated risk after querying node k
    """
    ## fu plus xk, yk = 0
    fu_xk0 = fu + (0 - fu[k])*Delta_uu_inv[:,k]/Delta_uu_inv[k,k]
    ## fu plus xk, yk = 1
    fu_xk1 = fu + (1 - fu[k])*Delta_uu_inv[:,k]/Delta_uu_inv[k,k]

    ## TODO ##

    return Rhat_fplus_xk
```

(5 points) Compute the above expected estimated risk for all unlabeled points and select one to query.

In [ ]:
```python
def zlg_query(Delta_uu_inv,n_l,fu,n_samples):
    """
    Args:
        Delta_uu_inv: Inverse matrix of the submatrix corresponding to unlabeled points.
        n_l: Number of labeled points.
        fu: Minimum energy solution of all unlabeled points.
        n_samples: Number of samples.
    Returns:
        query_idx: the idx of the point to query, wrt the unlabeled points
                (idx is 0 if it's the first unlabeled point)
    """
    n_u = n_samples - n_l
    query_idx = 0
    min_Rhat = np.inf
    ## TODO ##

    return query_idx
```

Let's try query 100 points. Which points are queried? Compare with random queries and make a plot.

In [ ]:
```python
n_samples = X.shape[0]
for t in range(100):
    ## edit this block ##
    query_idx = zlg_query(Delta_uu_inv,n_l,fu,n_samples)
    Yk = np.append(Yk,Yu[query_idx])
    Yu = np.delete(Yu,query_idx)
    Xk = np.append(Xk,[Xu[query_idx,:]],axis=0)
    Xu = np.delete(Xu,query_idx, 0)
    n_l += 1
    Delta = Laplacian_matrix(np.concatenate((Xk,Xu),axis=0))
    Delta_uu_inv, fu = minimum_energy_solution(Delta,n_l,Yk)
    print(query_idx)
    ## TODO ##
```

Bonus question (Your grade will not exceed 100 for this homework): For this dataset, how many labeled data points do you actually need, to train the model sufficiently well? And why?