Code principal : BGLR

```
import numpy as np
from scipy.linalg import eigh, pinv
from numpy.linalg import norm
import warnings
warnings.filterwarnings("ignore")

def normalize(X):
    return (X - np.mean(X, axis=0)) / (np.std(X, axis=0) + 1e-10)

def Lap(S):
    D = np.diag(np.sum(S, axis=1))
    return D - S

def get_Bipartite(X, h, mode=1):
    n = X.shape[1]
    idx = np.random.choice(n, h, replace=False)
    return X[:, idx]

def BGLR(X, k, m, h, beta, eta, maxIter):
    d, n = X.shape
    W = np.random.rand(d, m)
    P = np.zeros((n, h))
    lossFun = np.zeros(maxIter)
    E1 = np.zeros((n, n))
    E2 = np.zeros((h, h))
    I = np.eye(d)
    I1 = np.ones((d, d))
    di = np.zeros(n)

    featuresX = normalize(X.T)
    S1 = featuresX @ featuresX.T
    np.fill_diagonal(S1, 0)

    Z = get_Bipartite(X, h)

    for i in range(maxIter):
        print(f"--- Iteration {i+1} ---")
        print("Updating P...")
        di = np.zeros(n)
        for r in range(n):
            for l in range(m):
                di[r] += np.exp(-norm(W.T @ X[:, r] - W.T @ Z[:, l])**2 / eta)
            for j in range(h):
                P[r, j] = np.exp(-norm(W.T @ X[:, r] - W.T @ Z[:, j])**2 / eta) / (di[r] + np.finfo(
                    float).eps)

        print("Updating W...")
        new_X = np.concatenate((X, Z), axis=1)
```

```
        S = np.block([[E1, P], [P.T, E2]])
        L_S = Lap(S)
        A_X = new_X @ L_S @ new_X.T + beta * S1 @ I1
        A_X[np.isnan(A_X) | np.isinf(A_X)] = 1

        _, u = eigh(A_X)
        lambda_max = np.max(np.real(u))
        G = lambda_max * I - A_X

        if np.linalg.matrix_rank(G) <= m:
            Q1 = np.zeros((d, k))
            idx1 = np.argsort(np.diag(G))[::-1][:k]
            for s, b in enumerate(idx1):
                Q1[b, s] = 1
            M1 = Q1.T @ G @ Q1
            U1 = eigh(M1, eigvals=(M1.shape[0]-m, M1.shape[0]-1))[1]
            W = Q1 @ U1
        else:
            loss = np.zeros(20)
            for time in range(20):
                E = G @ W @ pinv(W.T @ G @ W) @ W.T @ G
                Q2 = np.zeros((d, k))
                idx2 = np.argsort(np.diag(E))[::-1][:k]
                for s, b in enumerate(idx2):
                    Q2[b, s] = 1
                M2 = Q2.T @ G @ Q2
                U2 = eigh(M2, eigvals=(M2.shape[0]-m, M2.shape[0]-1))[1]
                W = np.real(Q2 @ U2)
                loss[time] = np.trace(W.T @ E @ W)

        print("Calculating loss...")
        tem1 = beta * np.trace(W.T @ S1 @ I1 @ W)
        tem2 = np.trace(W.T @ new_X @ L_S @ new_X.T @ W)
        tem3 = eta * P * np.log(P + 1e-10)
        lossFun[i] = tem1 + tem2 + np.sum(tem3)

        if i > 10 and abs((lossFun[i] - lossFun[i - 1]) / lossFun[i]) < 1e-5:
            break

    index = np.where(W != 0)[0]
    newfea = X[index[:k], :]
    return W, index, newfea, lossFun
```

Exemple d'appel

```
X = np.random.rand(100, 50)
W, index, newfea, lossFun = BGLR(X, k=10, m=5, h=15, beta=0.1, eta=0.5, maxIter=30)
```