The code for reconstruction of DAG is submitted to <https://github.com/SplinterTao/DAG>

Use main.py to perform high-dimensional causal graph fitting with covariates, a modification of

<https://github.com/xunzheng/notears>, with application of drug discovery

Let n be sample size, d be dimension of X and d\* be dimension of

**Usage:** main\_fit(X, Z, lambda1, lambda2, loss\_type, max\_iter)

**Input:**

X: n by d Data matrix for the endogenous variables (e.g., gene expression data)

Z: Data matrix for the exogeneous variables (e.g., drug dosage)

lambda1 and lambda2: penalty parameters for endogenous and exogenous variables, respectively.

loss\_type: l1 and l2 losses are implemented

max\_iter: max number of iterations

**Output:** A d by d coefficient matrix for endogenous variables and a d by d\* matrix for covariates

Evaluation of fit: dagThreshold(G, threshold) transforms the fitted coefficient into a 0-1 matrix and isDAG(G) evaluates if the resulting matrix is DAG. If not*, the* fit is unsuccessful.

**Example:** Test performance for different hyperparameters

for hyper in [0.1,0.05,0.01,0.005,0.001]:

random.seed(10)

W\_est=notears\_linear(X,Y,lambda1=hyper,lambda2=hyper,loss\_type="l2",h\_tol=1e-8,w\_threshold=0.3)

W\_coef=W\_est[range(0,ngene),:]

W\_cova=W\_est[range(ngene,ngene+1),:]

W\_coef\_copy=W\_coef.copy()

for i in range(0,W\_coef\_copy.shape[0]):

W\_coef\_copy[i,i]=0

finalfit=findmin(W\_coef\_copy)[0]

NOA=sum(np.abs(finalfit.flatten()>0))

output1=pd.DataFrame(finalfit, columns =dataset.columns[range(1,len(dataset.columns)-1)].tolist(),index=dataset.columns[range(1,len(dataset.columns)-1)].tolist())

top10gene=np.flip(np.argsort(W\_cova\_reshape))[0:10]+1