TD #4

Large-scale Mathematical Programming

Leo Liberti, CNRS LIX Ecole Polytechnique
liberti@lix.polytechnique.fr

INF580

Universal Isometric Embedding

- Given metric space X with $|X| = n$ and distance matrix (DM) D
- UIE: finds embedding in ℓ_∞^n
- Define $x_{ik} = D_{ik}$ for all $i, k \leq n$
- **Thm.**: the DM of x is D

  proof seen in lecture
- Every graph $G = (V, E)$ gives rise to a metric space
  take $X = V$ and $d(u, v) =$ length of shortest path $u \to v$

Universal Isometric Embedding

Exercises (use AMPL and Python):

1. Generate random weighted biconnected graph G with $|V| = 50$
   output to AMPL .dat
2. Verify its connectedness using Floyd-Warshall's all-shortest-paths algorithm
3. Construct the DM $\bar{G}$ of the metric space induced by G
4. Find the UIE x of $\bar{G}$ in ℓ_∞
5. Verify the DM of x in ℓ_∞ is $\bar{G}$
6. Reduce the dimensionality of x to $K \in \{2, 3\}$ and draw the realization
   see below for PCA details

Principal Component Analysis

- PCA involves finding eigenvalues and eigenvectors
- AMPL can do it, but it's painful and inefficient
- Let's use Python instead

PCA: `dist2Gram`

```
## convert a distance matrix to a Gram matrix
def dist2Gram(D):
    n = D.shape[0]
    J = np.identity(n) - (1.0/n)*np.ones((n,n))
    G = -0.5 * np.dot(J,np.dot(np.square(D), J))
    return G
```

PCA: factor

```
## factor a square matrix
def factor(A):
    n = A.shape[0]
    (evals,evecs) = np.linalg.eigh(A)
    evals[evals < 0] = 0  # closest SDP matrix
    X = evecs
    sqrootdiag = np.eye(n)
    for i in range(n):
        sqrootdiag[i,i] = math.sqrt(evals[i])
    X = X.dot(sqrootdiag)
    # because default eig order is small->large
    return np.fliplr(X)
```

PCA: pca

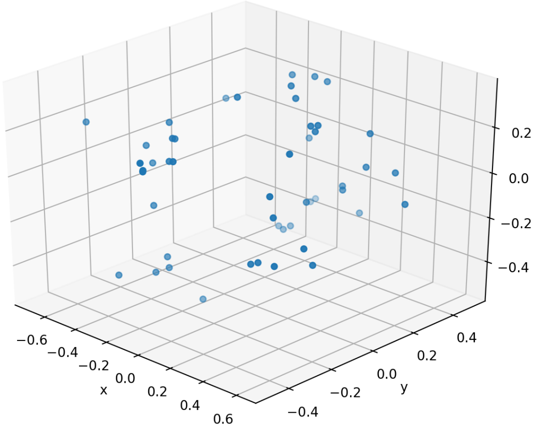
```
## principal component analysis
def PCA(B,K):
    x = factor(B)
    # only first K columns
    x = x[:,0:K]
    return x
```

PCA: `main`

```
import sys
import numpy as np
import math
import types
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
myZero = 1e-9
K = 3 # can be 2 or 3
f = sys.argv[1] # read input filename from command line
lines = [line.rstrip('\n').split()[2:] for line in open(f) if line[0] == 'x']
n = len(lines)
# turn into float array
X = np.array([[float(lines[i][j]) for j in range(n)] for i in range(n)])
G = dist2Gram(X) # if X produced by UIE, X = its own dist matrix
x = PCA(G,K)
if K == 2:
    plt.scatter(x[:,0], x[:,1])
elif K == 3:
    fig = plt.figure()
    ax = Axes3D(fig)
    ax.scatter(x[:,0], x[:,1], x[:,2])
plt.show()
```

PCA: exercise

Use Python code to display UIE of 50-vtx rnd graph in 3D



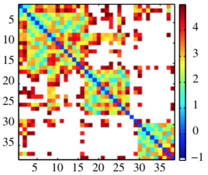
Distance Geometry Problem

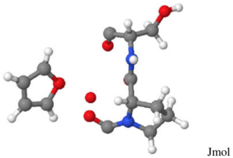
Use AMPL to implement 4 DGP MP formulations

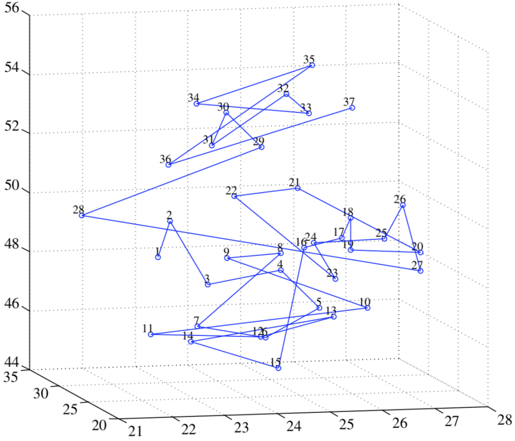
1. System of quadratic equations (sqp)):
   $\forall\{u,v\} \in E \quad \|x_u - x_v\|_2^2 = d_{uv}^2$
2. Slack/surplus variables (ssv):
   $\min\left\{ \sum_{\{u,v\}\in E} s_{uv}^2 \mid \forall\{u,v\} \in E \ \|x_u - x_v\|_2^2 = d_{uv}^2 + s_{uv} \right\}$
3. Unconstrained quartic polynomial (uqp):
   $\min \sum_{\{u,v\}\in E} (\|x_u - x_v\|_2^2 - d_{uv}^2)^2$
4. Pull-and-push (p&p):
   $\max\left\{ \sum_{\{u,v\}\in E} \|x_u - x_v\|_2^2 \mid \forall\{u,v\} \in E \ \|x_u - x_v\|_2^2 \le d_{uv}^2 \right\}$

and test them with the protein graph `tiny_gph.dat`

DGP: the `tiny_gph` instance

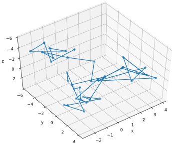
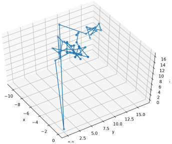
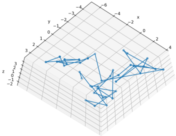






Distance Geometry Problem

- Use Python to draw the 4 realizations in 3D

sqp	ssv	uqp	p&p
none found			

are they similar?

- Compute the UIE of `tiny_gph.dat`

are there high values in UIE? Why?

- Use PCA to display it in 3D

with high values (left) / replace high values by -1.00 (right)