CMSE 820: Homework #10

Due on Nov 24, 2019 at 11:59pm

Professor Yuying Xie

Boyao Zhu

Problem 1

Solution// Define the local charts on $S^2 = \{(x, y, z) : x^2 + y^2 + z^2 = 1\}$ using: (1) polar coordinate representation; (2) stereographical projection.

Let $\Omega_1 = S^2 \setminus \{0, 0, 1\}$ and $\Omega_2 = S^2 \setminus \{0, 0, -1\}$ be two open sets in $\mathbb{R}$. Then $C = \{\Omega_i : i \in \{1, 2\}\}$ is an open cover of S^2.
Applying stereographical projection for these two open sets, we can define the following mappings

$$\begin{aligned}
\phi_1 : \Omega_1 \to \mathbb{R}^2, \phi_1(x, y, z) &= \left(\frac{x}{1-z}, \frac{y}{1-z}\right) \\
\phi_1^{-1} : \mathbb{R}^2 \to \Omega_1, \phi_1^{-1}(X, Y) &= \left(\frac{2X}{1+X^2+Y^2}, \frac{2Y}{1+X^2+Y^2}, \frac{-1+X^2+Y^2}{1+X^2+Y^2}\right) \\
\phi_2 : \Omega_2 \to \mathbb{R}^2, \phi_2(x, y, z) &= \left(\frac{x}{1+z}, \frac{y}{1+z}\right) \\
\phi_2^{-1} : \mathbb{R}^2 \to \Omega_2, \phi_1^{-1}(X, Y) &= \left(\frac{2X}{1+X^2+Y^2}, \frac{2Y}{1+X^2+Y^2}, \frac{1-X^2-Y^2}{1+X^2+Y^2}\right)
\end{aligned}$$

It is easy to check that ϕ_1 and ϕ_2 are homeomorphisms, so they are two local charts under the stereographical projection that form an atlas of S^2.
Rewriting the Cartesian coordinate representation of S^2 in the polar coordinate representation, we have a different set of local charts.

$$\begin{aligned}
\psi_1 : \Omega_1 \to [0, \infty) \times [0, 2\pi), \psi_1(x, y, z) &= \left(\frac{\sqrt{x^2+y^2}}{1-z}, \text{arccos}\frac{y}{x}\right) \\
\psi_1^{-1} : [0, \infty) \times [0, 2\pi) \to \Omega_1, \psi_1^{-1}(r, \theta) &= \left(\frac{2r\text{cos}\theta}{1+r^2}, \frac{2r\text{sin}\theta}{1+r^2}, \frac{r^2-1}{1+r^2}\right) \\
\psi_2 : \Omega_2 \to [0, \infty) \times [0, 2\pi), \psi_2(x, y, z) &= \left(\frac{\sqrt{x^2+y^2}}{1+z}, \text{arccos}\frac{y}{x}\right) \\
\psi_2^{-1} : [0, \infty) \times [0, 2\pi) \to \Omega_2, \psi_2^{-1}(r, \theta) &= \left(\frac{2r\text{cos}\theta}{1+r^2}, \frac{2r\text{sin}\theta}{1+r^2}, \frac{-r^2+1}{1+r^2}\right)
\end{aligned}$$

Problem 2

Solution

Kernel PCA is only able to separate the data points into 3 rays of similar color, which still differs quite a great deal from the original data. LLE, on the other hand, is capable of "unrolling" the data, restoring the 2D-manifold features (almost perfectly). When the high-dimensional data resembles some low-dimensional manifold embedded in the high- dimensional space, LLE outperforms plain kernel PCA.

Problem 2 continued on next page. . .

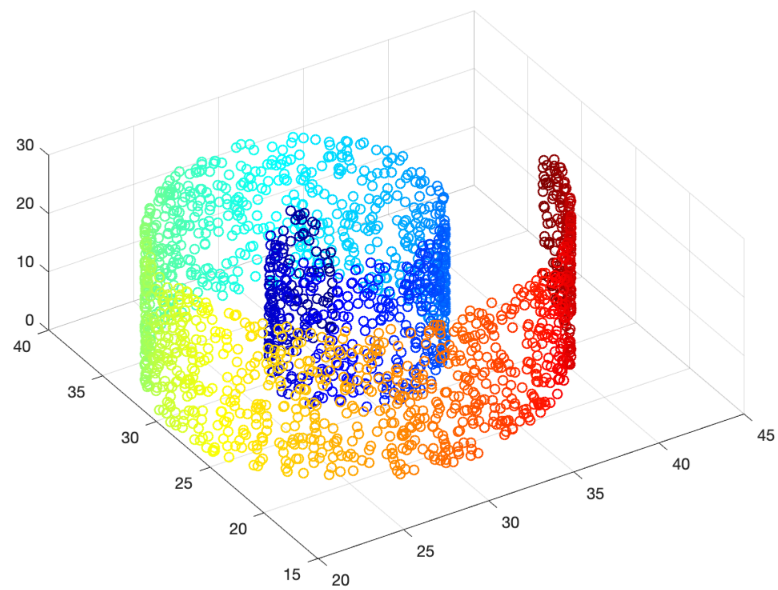

Figure 1: Original data in 3D

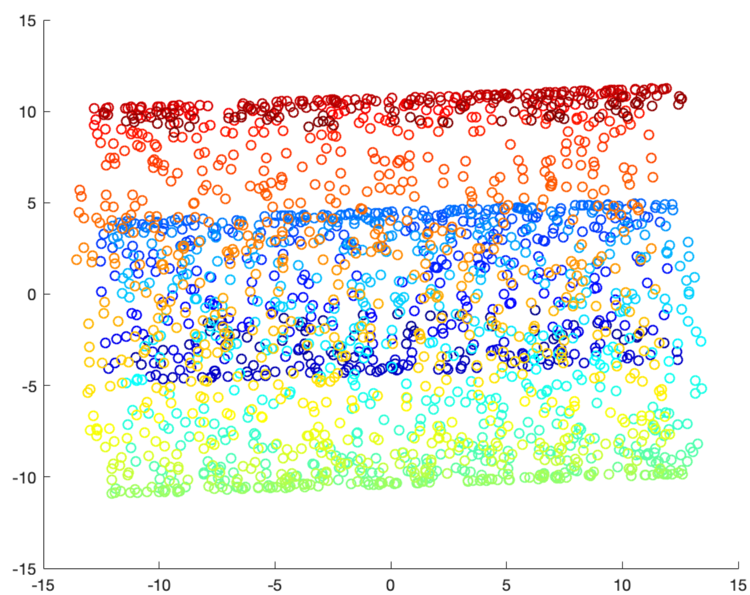

Figure 2: By PCA, Data projected onto the top 2 PCs

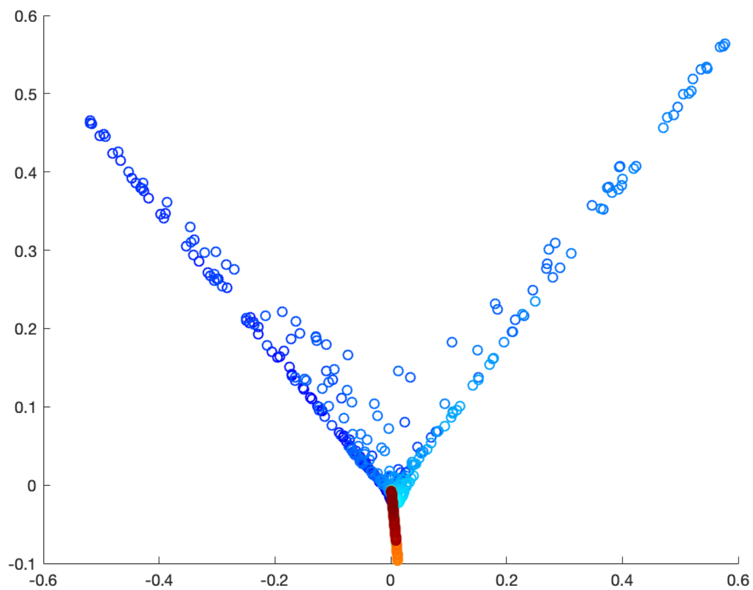

Figure 3: By Kernel PCA, Data projected on the top 2 PCs

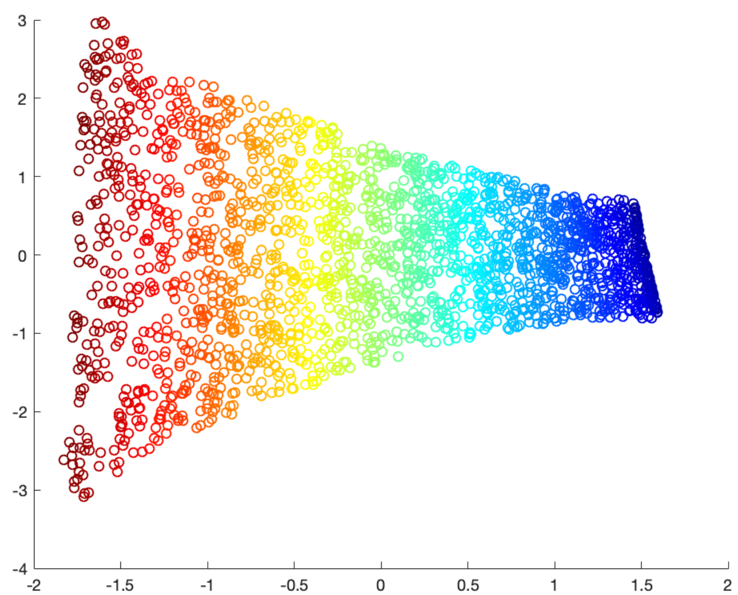

Figure 4: LLE with 12-nearest neighbors, Data reduced to 2D

Problem 3

Solution
Laplacian Eigenmaps (LE).
Let $x_1, \cdots, x_n \in \mathbb{R}^p$ be the high dimensional data points. For any fixed scalar $\epsilon > 0, x_i$ and x_j are called ϵ-neighbors of each other if and only if $\|x_i - x_j\|_2 \leq \epsilon$. Fix another scalar $\sigma^2 > 0$, and we can define a weight $w_{ij} = \exp(-\frac{\|x_i - x_j\|_2^2}{\sigma^2})$ if x_i and x_j are ϵ-neighbors and $w_{ij} = 0$ otherwise. A reasonable low dimensional embedding $y_1, \cdots, y_n$ minimizes the following objective function

$$\sum_{i,j} w_{ij}\|y_i - y_j\|_2^2.$$

3.1 a

Prove that $\min\frac{1}{2}\sum w_{ij}\|y_i - y_j\|_2^2 = \min Tr(YLY^T$, where $Y = [y_1, \cdots, y_n]$ and $L = D - A$ with $A_{ij} = w_{ij}$ and D being a diagonal matrix with $D_{ii} = \Sigma_j w_{ij}$. The matrix L is called graph laplacian.
Proof: Starting from the summation, we have

$$\begin{aligned}
\frac{1}{2}\sum_{ij} w_{ij}\|y_i - y_j\|_2^2 &= \frac{1}{2}\sum_{ij} w_{ij}(y_i^T y_i - y_i^T y_j - y_j^T y_i + y_j^T y_j) \\
&= \frac{1}{2}[2\sum_i(\sum_j w_{ij})y_i^T y_i - 2\sum_{ij} w_{ij} y_i^T y_j] \\
&= \sum_i(\sum_j w_{ij})y_i^T y_i - \sum_{ij} w_{ij} y_i^T y_j
\end{aligned}$$

Starting from the trace, we have

$$\begin{aligned}
\mathrm{Tr}(YLY^T) &= \sum_i (YLY^T)_{ii} \\
&= \sum_i[\sum_{jk} Y_{ij} L_{jk} (Y^T)_{ki}] \\
&= \sum_i(\sum_{jk} Y_{ij} L_{jk} Y_{ik}) \\
&= \sum_{jk} L_{jk}(\sum_i Y_{ij} Y_{ki} \\
&= \sum_{jk} L_{jk} y_j^T y_k \\
&= \sum_{ij}(D_{ij} - A_{ij}) y_i^T y_j \\
&= \sum_{ij}(\sum_k w_{ik})\delta_{ij} y_i^T y_j - \sum_{ij} w_{ij} y_i^T y_j \\
&= \sum_i(\sum_j w_{ij}) y_i^T y_i - \sum_{ij} w_{ij} y_i^T y_j \\
&= \frac{1}{2}\sum_{ij} w_{ij}\|y_i - y_j\|_2^2
\end{aligned}$$

3.2 b

$$\hat{Y} = \arg\min_Y \mathrm{Tr}(YLY^T) \text{ subject to } YDY^T = I \text{ and } YD\mathbf{1} = 0$$

Show that the solution $\hat{Y} \in \mathbb{R}^{d\times n}$ are given by the eigenvectors corresponding to the lowest d eigenvalues of the generalized eigenvalue problem

$$Ly = \lambda Dy$$

Problem 3 continued on next page. . .

Proof: We ignore the constraint $YD\mathbf{1} = 0$ for the time being and write down the Lagrangian

$$L(Y, \Lambda) = \text{Tr}(YLY^T) + \langle \Lambda, I - YDY^T \rangle$$

where Λ is the symmetric Lagrangian matrix coefficients, because YDY^T is symmetric. Taking the derivative respect to Y, we have

$$\frac{\partial L}{\partial Y} = 2(LY^T - \Lambda DY^T)$$

Setting the derivative to be 0 and rewriting $\Lambda = U\Sigma U^T$ in terms of its eigenvalue decomposition with Σ beging diagonal and U being orthogonal,

$$LY^T - \Lambda DY^T = LY^T - U\Sigma U^T DY^T = LY^T - U\Sigma DU^T Y^T = 0 \Longrightarrow LY' = DY'\Sigma \quad (*)$$

where $Y' = (UY)^T$ and we have used the fact that matrix multiplication with any diagonal matrix is commutative.
Note that for any Y and any orthogonal matrix O, $\text{Tr}(YLY^T) = \text{Tr}((OY)L(OY)^T)$. $\hat{Y}$ can be determined at most up to rotations. This suggests that we should solve the generalized eigenvalue problem

$$Ly = \lambda Dy \quad (**)$$

Furthermore, we should show that for any y^* satistfying equation $(**)$, $\mathbf{1}^T Dy^* = 0$

$$\begin{aligned} & \mathbf{1}^T Dy^* = \mathbf{1}^T \frac{1}{\lambda} Ly = 0 \\ \Longleftarrow \quad & \mathbf{1}^T L = 0^T \\ \Longleftarrow \quad & \forall j, \sum_j L_{ij} = \sum_j D_{ij} - A_{ij} = D_{ii} - \sum_j W_{ij} = \sum_j W_{ij} - \sum_j W_{ij} = 0 \end{aligned}$$

So we can let $\hat{Y} = [y_1, \cdots, y_d]^T$ such that $y_1, \cdots, y_d$ are solutions to the generalized eigenvalue problem corresponding to the smallest eigenvalues. It is clear that $\hat{Y} D \hat{Y}^T = I$ and $\hat{Y} D\mathbf{1} = 0$

Problem 4

Solution
In the derivation of LLE, we define

$$M = (I_N - W)^T (I_N - W)$$

Prove that $M\mathbf{1} = 0$
Proof Recall that

$$\forall i, \sum_j W_{ij} = W_i, \mathbf{1} = 1$$

It follows that

$$W\mathbf{1} = \mathbf{1}$$

Hence,

$$M\mathbf{1} = (I_N - W)^T (I_N - W)\mathbf{1} = (I_N - W)^T (\mathbf{1} - \mathbf{1}) = 0$$

```
M = csvread('HW10_dat.csv',1);
Color = csvread('HW10_color.csv');
X = M';

% Original data
figure;
scatter3(X(1,:),X(2,:),X(3,:),36,Color);
%
% % PCA
n = size(M,1);
mean = sum(M,1)/n;
PC = pca(M);
for i = 1:n
    proj(i,1) = (M(i,:)-mean)*PC(:,1);
    proj(i,2) = (M(i,:)-mean)*PC(:,2);
end
figure;
scatter(proj(:,1),proj(:,2),36,Color);
%
% % Gaussian KPCA
KK = KappaMatrix(X,'GaussianKernel',1.3);
[Sigma2,V2,Y2]=KernelPCA(KK,2);
figure;
scatter(Y2(1,:),Y2(2,:),36,Color);

k = 12;
scaling = sqrt(n);
Y = LocallyLinearEmbedding(X,k,scaling);
size(Y);
figure;
scatter(Y(:,1),Y(:,2),36,Color);


function D = DistanceMatrix(X)
    n = size(X,2);
    p = size(X,1);
    D = zeros(n,n);
    for i = 1:n
        D(i,i) = 0;
        xi = X(:,i);
        for j = i+1:n
            xj = X(:,j);
            D(i,j) = norm(xi-xj);
            D(j,i) = D(i,j);
        end
    end
end

function index = kNearestNeighbours(D,k)
    n = size(D,1);
    ind = zeros(k+1);
```

```
    index = zeros(n,k);
    for i = 1:n
        [~,ind] = mink(D(i,:),k+1);
        index(i,:) = ind(2:k+1);
    end
end

function W = weight(X,index)
    n = size(X,2);
    k = size(index,2);
    C = zeros(k,k);
    kones = ones(k,1);
    Wt = zeros(k,n);
    for i = 1:n
        xi = X(:,i);
        for p = 1:k
            xp = X(:,index(i,p));
            for q = p:k
                xq = X(:,index(i,q));
                C(p,q) = (xi-xp)'*(xi-xq);
                C(q,p) = C(p,q);
            end
        end
        s = svd(C);
        C = C + trace(C)*eye(k)/1000;
        temp = C\kones;
        Wt(:,i) = temp/(kones'*temp);
    end
    W = zeros(n,n);
    for i = 1:n
        for j = 1:k
            W(i,index(i,j)) = Wt(j,i);
        end
    end
end

function Y = ConstructY(W,k,scaling)
    n = size(W,1);
    I = eye(n);
    M = (I-W)'*(I-W);
    [U,S,~] = svd(M);
    for i = n:-1:1
        if(S(i,i) > S(n,n))
            target = i;
            break;
        end
    end
    Y = zeros(n,k);
    for i = 1:k
        Y(:,i) = U(:,target)*scaling;
        target = target-1;
```

```
    end
end

function Y = LocallyLinearEmbedding(X,k,scaling)
    D = DistanceMatrix(X);
    index = kNearestNeighbours(D,k);
    W = weight(X,index);
    Y = ConstructY(W,k,scaling);
end

% Taken from HW5.m

function G = GaussianKernel(x,y,sigma)
    G = exp(-((norm(x-y)/sigma)^2)/2);
end

function K = KappaMatrix(X,type,tuning_parameter)
  p = size(X,1);
  n = size(X,2);
  H = eye(n)-ones(n,1)*ones(1,n)/n;
  if type == "Polynomial"
      X = X*H;
  end
  K = zeros(n,n);
  for i=1:n
      xx = X(:,i);
      for j=i:n
         yy = X(:,j);
         if type == "GaussianKernel"
           K(i,j)=GaussianKernel(xx,yy,tuning_parameter);
         elseif type == "Polynomial"
           K(i,j)=PolynomialKernel(xx,yy,tuning_parameter);
         end
         K(j,i)=K(i,j);
      end
  end
  K = H*K*H;
end

function [Sigma,V,Y] = KernelPCA(kappa,d)
    [U,S,V] = svd(kappa);
    V = V(:,1:d);
    Sigma = sqrtm(S(1:d,1:d));
    Y = Sigma*V.';
end
```