Dimension Reduce 初步学习小结

对于一些标准的数据格式（每个数据对应欧几里德空间里的一个向量），那么降维的过程，就是把这些原本高维数，难以处理的数据映射到低维，易分析和处理的过程。就像是一个黑盒子，输入的是 D*N 的矩阵，输出的是 d*N 的矩阵(d<D)。当然，在降维压缩数据的同时，我们不可避免会造成信息的损失。然后，这也不是一个绝对有害的问题。只要找到合适的降维算法，使得损失的信息量更多的偏向于噪声，而非我们关注的 Feature，便能够去其糟粕，取其精华，将无用信息丢弃的时候，最大限度的保留有用的 Features.

那么，怎么样去评定一个降维过程的好坏呢？我们通过定义一个函数

error function：$\frac{1}{N}\sum_1^N \|Xi - \widehat{Xi}\|^2$

其中$\widehat{Xi}$是 Xi 对应的低维形式重新构造出来的高维形式。上式的意义就相当于使得压缩后解压出来的数据损失量，因此我们令上式最小化。

关于 Dimension Reduce 的宏观概念，pluskid 已经讲得很生动了，今后复习时可以参考（http://blog.pluskid.org/?p=290），主要实在这里结合 DengCai 的代码，初略的总结下一些典型降维算法的计算过程，至于具体更加形象化的对比，得等到自己真正把项目做起来后，才能更准确地比较出来。

PCA：

PCA 的建立理论基础，就是在信号处理中，通常认为信号具有较大的方差，噪声有较小的方差，信噪比就是信号与噪声的方差比，越大越好。于是，有最大方差理论和最小平方误差两个理论给予佐证（http://www.cnblogs.com/jerrylead/archive/2011/04/18/2020209.html）。这里只从最大方差理论出发给以推导：(这里设投影后均值为 0，u'为投影向量)

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^{m}(x^{(i)T}u)^2 &= \frac{1}{m}\sum_{i=1}^{m}u^T x^{(i)}x^{(i)T}u \\ &= u^T\left(\frac{1}{m}\sum_{i=1}^{m}x^{(i)}x^{(i)T}\right)u.\end{aligned}$$

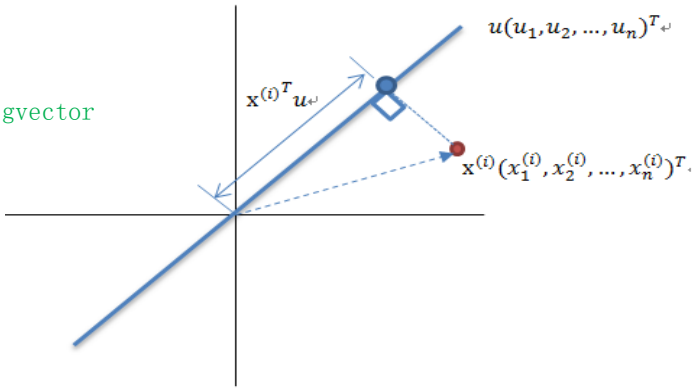
即 Arg max u^TXX^Tu。如果设约束条件为u^Tu=1，设最大值为ϰ，那么可等价于

$$u^TXX^Tu=\varkappa u^Tu \quad => \quad XX^Tu=\varkappa u \quad [Cu=\lambda u]$$

因而，最大方差值便是上式的最大特征值，最佳投影直线便是最大特征值对应的对大特征向量。其次ϰ是第二大对应的特征向量，依次类推。

核心代码：

```
data = (data - repmat(sampleMean,nSmp,1));
[eigvector, eigvalue] = mySVD(data',ReducedDim);
% [U,S,V] = mySVD(X,ReducedDim)
%  X = U*S*V'.
%  等价于
%  data' data*eigvector=eigvalue*eigvector
% 详细解释见下方NPE处
```



KPCA：

由于 PCA 算法先验条件是样本点满足高斯分布，顾其高度依赖于样本点的均值和方差，对于不满足该分布的数据集，往往不能很好的进行分析。但是，我们可以假设数据潜在含有更高维数的高斯分布，可以在更高维空间（Hibert Space）中进行 PCA 分析。因而，我们引入一个映射Φ：X->F，这里 F 表示的是 Hibert 泛函空间。设$K_{ij}=\Phi_i^T\Phi_j$，

令$\Phi_i=\Phi(x_i)$，居中$\Phi_i^C=\Phi_i-\frac{1}{N}\sum_k\Phi_k$

$$
\begin{aligned}
K_{ij}^C &=<\Phi_i^C\Phi_j^C> \\
&=(\Phi_i-\frac{1}{N}\sum_k\Phi_k)^T(\Phi_j-\frac{1}{N}\sum_l\Phi_l) \\
&=\Phi_i^T\Phi_j-\frac{1}{N}\sum_l\Phi_i^T\Phi_l-\frac{1}{N}\sum_k\Phi_k^T\Phi_j+\frac{1}{N^2}\sum_k\sum_l\Phi_k^T\Phi_l \\
&=K_{ij}-\frac{1}{N}\sum_l K_{il}-\frac{1}{N}\sum_k K_{kj}+\frac{1}{N^2}\sum_k\sum_l K_{kl}
\end{aligned}
$$

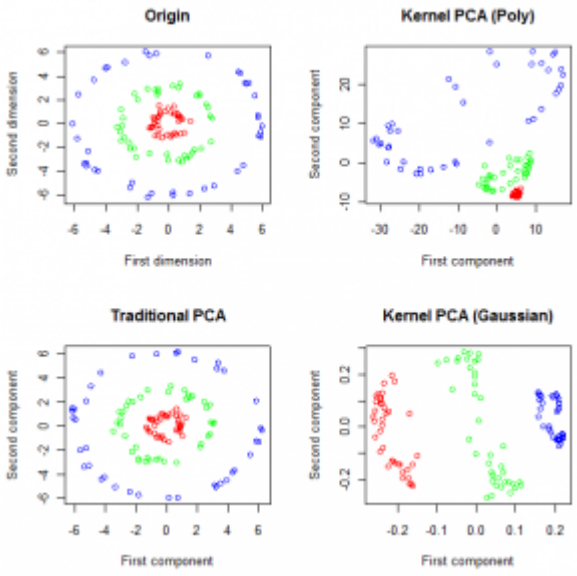
与传统 PCA 同理：$K^C u=\lambda u$;

核心代码：

```
D = EuDist2(fea_a,[],0);
K = exp(-D/(2*options.t^2));%高斯核
sumK = sum(K,2);
H = repmat(sumK./nSmp,1,nSmp);
Kc = K - H - H' + sum(sumK)/(nSmp^2);
K = max(K,K');
[eigvector, eigvalue] = eig(K);
```

其中EuDist2:

```
aa = sum(fea_a.*fea_a,2);
bb = sum(fea_b.*fea_b,2);
ab = fea_a*fea_b';
D = bsxfun(@plus,aa,bb') - 2*ab;
```



对于新的数据，同样可以：

$$
\begin{aligned}
K(x_i,t)^C &=<\Phi_i^C\Phi_t^C> \\
&=(\Phi_i-\frac{1}{N}\sum_k\Phi_k)^T(\Phi_t-\frac{1}{N}\sum_l\Phi_l) \\
&=\Phi_i^T\Phi_t-\frac{1}{N}\sum_l\Phi_i^T\Phi_l-\frac{1}{N}\sum_k\Phi_k^T\Phi_t+\frac{1}{N^2}\sum_k\sum_l\Phi_k^T\Phi_l \\
&=K(x_i,t)-\frac{1}{N}\sum_l K_{il}-\frac{1}{N}\sum_k K(x_k,t)+\frac{1}{N^2}\sum_k\sum_l K_{kl}
\end{aligned}
$$

相应代码：

```
Ktest = constructKernel(feaTest,fea,options)%Kij
Y = Ktest*eigvector;
```

LE:

如果我们不知道数据的具体特征，只知道数据之间的相似程度，那么降维算法就需要保证在原本空间中的相似的数据在映射后空间中仍然相似。LE 从图的角度去构建数据之间的相似性，每个数据代表途中的一个点，每条边代表数据相似度的一个权重。

Step1:计算确定每个点的领域（根据 Knn 或 ε -neighbors 原则），Dij

Step2:选择权值类型并从领域内点计算权重：$W_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}}$

Step3:计算领域点之间的权重：argmin $\sum_{ij}(y_i - y_j)^2 W_{ij}$

$$\sum_{i,j}(y_i - y_j)^2 W_{ij} = \sum_{i,j}(y_i^2 + y_j^2 - 2y_i y_j)W_{ij} =$$

$$\sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2\sum_{i,j} y_i y_j W_{ij} = 2\mathbf{y}^T L\mathbf{y}$$

$$\underset{\mathbf{y}}{\operatorname{argmin}}\ \mathbf{y}^T L\mathbf{y}$$
$$\mathbf{y}^T D\mathbf{y}=1$$

由于 L=D-W 上式等价于 argmax $\boldsymbol{y^T W y}$ with $\boldsymbol{y^T y=1}$，即 $\mathbf{Wy= \lambda y}$

核心代码:

```
#constructW
    dist = EuDist2(fea(smpIdx,:),fea,0);
    nSmpNow = length(smpIdx);
    dump = zeros(nSmpNow,options.k+1);
    idx = dump;
    for j = 1:options.k+1
        [dump(:,j),idx(:,j)] = min(dist,[],2);
         temp = (idx(:,j)-1)*nSmpNow+[1:nSmpNow]';
         dist(temp) = 1e100;
    end
G((i-1)*BlockSize*(options.k+1)+1:nSmp*(options.k+1),1)=
    repmat(smpIdx',[options.k+1,1]);
G((i-1)*BlockSize*(options.k+1)+1:nSmp*(options.k+1),2)=idx(:);
G((i-1)*BlockSize*(options.k+1)+1:i*BlockSize*(options.k+1),3) =
dump(:);
W = sparse(G(:,1),G(:,2),G(:,3),nSmp,nSmp);
W = W - diag(diag(W));%去掉自身情况，即W（i,i）=0,W(i,i)消失在sparse中#CW
data = (data - repmat(sampleMean,nSmp,1));
[eigvector, eigvalue] = eig(W);%Y= eigvector(top d)
```

LLE:

LLE 假设数据分布在 manifold 上,利用局部欧式空间的特性，进行领域范围的点的线性组合，进而形成全局的非线性降维，同时尽可能保持其原有分拓扑关系.同时， LE 仅假设数据与离他最近的点相似，而 LLE 假设数据可以用领域的点线性表示(Step2 目标函数因此不同)：

Step1:计算确定每个点的领域集合（根据 Knn 或 ε -neighbors 原则），的 neighbors；

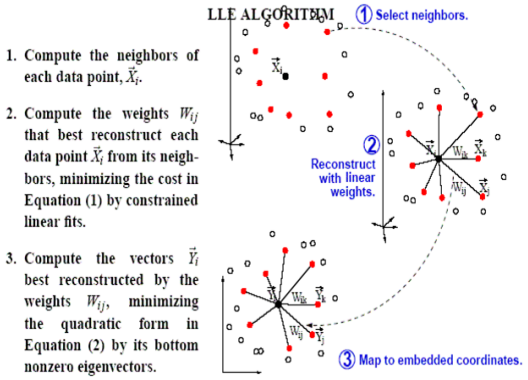
Step2:计算领域点之间的权重： $\min \sum_i \|\mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j\|^2$ with $\sum_j W_{ij} = 1, j = 1, 2, \ldots, m$

$C_{jk} = (\vec{x} - \vec{\eta}_j) \cdot (\vec{x} - \vec{\eta}_k).$

于是 $\varepsilon = \left|\vec{x} - \sum_j w_j \vec{\eta}_j\right| = \left|\sum_j w_j (\vec{x} - \vec{\eta}_j)\right|^2 = \sum_{jk} w_j w_k C_{jk},$

结合$\sum Wij = 1$，拉格朗日乘数法得

$$w_j = \frac{\sum_k C_{jk}^{-1}}{\sum_{lm} C_{lm}^{-1}}.$$



由此得到最佳权重矩阵，确保线性组合造成的损失最小。

Step3. 求解映射:Argmmin $\Phi(\mathbf{y}) = \sum_i \left(y_i - \sum_j W_{ij} y_j\right)$

设 $M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum W_{ki} W_{kj}$，于是上式等价于：

$\Phi(Y) = \sum_{ij} M_{ij}(\vec{Y}_i \cdot \vec{Y}_j) = \lambda$

若约束为 $\frac{1}{N}\sum_i \vec{Y}_i \vec{Y}_i^{\top} = I$

则 My=λy,即 y 为 smallest 2~d+1 维特征向量。

核心代码:

```
X2 = sum(X.^2,1);
distance = repmat(X2,N,1)+repmat(X2',1,N)-2*X'*X;
[sorted,index] = sort(distance);
 neighborhood = index(2:(1+K),:);
W = zeros(K,N);
 for ii=1:N
   z = X(:,neighborhood(:,ii))-repmat(X(:,ii),1,K);
   C = z'*z; % local covariance
   C = C + eye(K,K)*tol*trace(C);  % regularlization (K>D)
   W(:,ii) = C\ones(K,1);  % ====solve Cw=1
   W(:,ii) = W(:,ii)/sum(W(:,ii));  % enforce sum(w)=1
end;
for ii=1:N   w = W(:,ii);
   jj = neighborhood(:,ii);
   M(ii,jj) = M(ii,jj) - w';
   M(jj,ii) = M(jj,ii) - w;
   M(jj,jj) = M(jj,jj) + w*w';
end;
[Y,eigenvals] = eigs(M,d+1,0,options);%2000*2000
Y = Y(:,2:d+1)'*sqrt(N);
```

NPE：

NPE 是 LLE 的一种线性近似，它保持了流体的局部结构，并且可以 supervise 和 unsupervise.

Step1:计算确定每个点的领域集合（根据 Knn 或 ε -neighbors 原则），得 neighohoods；

Step2:计算领域点之间的权重： $\min \sum_i \|\mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j\|^2$,with $\sum_j W_{ij} = 1, j = 1, 2, \ldots, m$(See LLE)

Step3.求解映射:Argmin

设z_i=y_i-$\sum W_{ij}y_i$, $\Phi(\mathbf{y}) = \sum_i \left(y_i - \sum_j W_{ij} y_j \right)^2$

z=y-Wy=(I-W)y

$\Phi(y)$=$\sum(y_i - \sum W_{ij}y_i)$

=$\sum(z_i)^2$

=z^Tz

=y^T（I − W)^T(I-W)y

=a^TX（I − W)^T(I-W) X^Ta

=a^TXMX^Ta

设约束$y^Ty = 1$，即$a^TXX^Ta = 1$,得

Argmin a^TXMX^Ta with $a^TXX^Ta = 1$

等价于a^TXMX^Ta=≺ a^TXX^Ta的求解.

若 X 的列向量线性独立，则XX^T是奇异矩阵（不可逆）,即不能直接求解特征向量，所以采用 SVD（奇异值分解）,令 X=US V^T（其中 U<- XX^T,V<-X^TX,S<-≺,且均满秩正交）.

上式可表达为：

$a^TUSVMVS'U'a$=≺ $a^TUSVVS'Ua$ X=USV'

b'VMV'b=≺b'b; X'=VSU'

VMV'b=≺b; V=X'US^{-1}

a=US^{-1}b

核心代码:

```
for idx=1:nLabel
   classIdx = find(options.gnd==Label(idx));
   Distance = EuDist2(data(classIdx,:),[],0);
   [sorted,index] = sort(Distance,2);
   neighborhood(classIdx,:) = classIdx(index(:,2:(1+options.k)));
End
```

求W,M方法同LLE;

```
LGE: [eigvector, eigvalue] = LGE(M, D, options, data)
MySVD: [U, S, V] = mySVD(data)
ddata = X*X'; [U, eigvalue] = eig(ddata)
eigvalueHalf = eigvalue.^.5;
S=spdiags(eigvalueHalf,0,length(eigvalueHalf),length(eigvalueHalf));
eigvalue_MinusHalf = eigvalue_Half.^-1;
V = X'*(U.*repmat(eigvalue_MinusHalf',size(U,1),1));%V<-U
END MySVD
data=U; eigvalue_PCA=full(diag(S));
eigvector_PCA=V*spdiags(eigvalue_PCA.^-1,0,length(eigvalue_PCA),
length(eigvalue_PCA));
WPrime = data'*M*data;
[eigvector, eigvalue] = eig(WPrime);
eigvector = eigvector_PCA*eigvector;
END LGE
```

LPP:

LPP 是 LE 的线性近似，又与 NPE 一样是局部保留的线性映射（目标函数不同）。

Step1:计算确定每个点的领域（根据 Knn 或 ε -neighbors 原则），Dij;

Step2:选择权值类型并从领域内点计算权重：$W_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}}$ ；

Step3:计算领域点之间的权重：min $\sum (y_i - y_j)^2 W_{ij}$,with $\mathbf{y}^T D\mathbf{y} = 1 \Rightarrow \mathbf{a}^T XDX^T\mathbf{a} = 1$

$$\frac{1}{2}\sum_{ij}(y_i - y_j)^2 W_{ij} = \frac{1}{2}\sum_{ij}(\mathbf{a}^T\mathbf{x}_i - \mathbf{a}^T\mathbf{x}_j)^2 W_{ij}$$

即(D-W)y=⊀Dy $= \sum_i \mathbf{a}^T\mathbf{x}_i D_{ii}\mathbf{x}_i^T\mathbf{a} - \sum_{ij}\mathbf{a}^T\mathbf{x}_i W_{ij}\mathbf{x}_j^T\mathbf{a} = \mathbf{a}^T X(D-W)X^T\mathbf{a} = \mathbf{a}^T XLX^T\mathbf{a}$

=>Wy=(1-⊀)Dy

=> $WD^{-1/2}D^{1/2}X'a$ =(1-⊀)$D^{1/2}D^{1/2}X'a$

=>$D^{1/2}$ $WD^{-1/2}D^{1/2}X'a$=(1-⊀)$D^{1/2}X'a$

Argmin a^TXLX^Ta <=>argmax $a^T D^{1/2}$ $WD^{-1/2}D^{1/2}X'a$ with $a^T XDX^T$a=1

⇨ $\hat{X}^T D^{1/2}$ $WD^{-1/2}\hat{X}a$ =λ$\hat{X}^T\hat{X}$a 其中$\hat{X} = D^{1/2}X'$

核心代码:

```
#constructW(CW)
    dist = EuDist2(fea(smpIdx,:),fea,0);
    nSmpNow = length(smpIdx);
    dump = zeros(nSmpNow,options.k+1);
    idx = dump;
    for j = 1:options.k+1
        [dump(:,j),idx(:,j)] = min(dist,[],2);
         temp = (idx(:,j)-1)*nSmpNow+[1:nSmpNow]';
         dist(temp) = 1e100;
    end
G((i-1)*BlockSize*(options.k+1)+1:nSmp*(options.k+1),1)=
    repmat(smpIdx',[options.k+1,1]);
G((i-1)*BlockSize*(options.k+1)+1:nSmp*(options.k+1),2)=idx(:);
G((i-1)*BlockSize*(options.k+1)+1:i*BlockSize*(options.k+1),3) =
dump(:);
W = sparse(G(:,1),G(:,2),G(:,3),nSmp,nSmp);
W = W - diag(diag(W));%去掉自身情况，即W（i,i）=0,W(i,i)消失在sparse中#CW
data = (data - repmat(sampleMean,nSmp,1));
D = full(sum(W,2));%Dij
DToPowerHalf = D.^.5;
D_mhalf = DToPowerHalf.^-1;
    tmpD_mhalf = repmat(D_mhalf,1,nSmp);
    W = (tmpD_mhalf.*W).*tmpD_mhalf';
data = repmat(DToPowerHalf,1,nFea).*data;% X̂
[eigvector, eigvalue] = LGE(W, [], options, data);
#LGE 处理argmax aX'WXa=λaX'Xa <= X̂^T D^{1/2} WD^{-1/2}X̂a =λX̂^T X̂a
[U, S, V] = mySVD(data); X = U*S*V'.
[U, S, V]=CutonRatio(U,S,V,options);
data = U;%PHD PAPER P19
eigvector_PCA=V*spdiags(eigvalue_PCA.^-1,0,length(eigvalue_PCA),lengt
h(eigvalue_PCA));
WPrime = data'*W*data;%X'*M*X
WPrime = max(WPrime,WPrime')
[eigvector, eigvalue] = eig(WPrime);
eigvector = eigvector(:, 1:ReducedDim);
eigvalue = eigvalue(1:ReducedDim);
eigvector = eigvector_PCA*eigvector;#END LGE
```

KLPP：

我们可以假设数据潜在含有更高维数的高斯分布，可以在更高维空间（Hibert Space）中进行 PCA 分析。因而，我们引入一个映射 Φ：X->H，这里 H 表示的是 Hibert 泛函空间。

设$K_{ij}= \Phi_i^T \Phi_j$

则根据 LPP，可得$[\phi(X)L\phi^T(X)]\nu = \lambda[\phi(X)D\phi^T(X)]\nu$

Argmin Φ(X)LΦ(X)'v=≮Φ(X)DΦ(X)'v

Argmax Φ(X)WΦ(X)'v=≮Φ(X)DΦ(X)'v

=>令 Φ(X)=USV'　其中 U<-Φ(X)c'Φ(X)c<->eig(Kc)

vUSV'WVSU'v=≮vUSV'DVSU'v;

b'VWV'b=≮b'VDVb;

v=US^{-1}b

令$\Phi_i = \Phi(x_i)$, 居中$\Phi_i^C = \Phi_i - \frac{1}{N}\sum_k \Phi_k$

$$K_{ij}^C = <\Phi_i^C \Phi_j^C>$$

$$= (\Phi_i - \frac{1}{N}\sum_k \Phi_k)^T(\Phi_j - \frac{1}{N}\sum_l \Phi_l)$$

$$= \Phi_i^T\Phi_j - \frac{1}{N}\sum_l \Phi_i^T\Phi_l - \frac{1}{N}\sum_k \Phi_k^T\Phi_j + \frac{1}{N^2}\sum_k\sum_l \Phi_k^T\Phi_l$$

$$= K_{ij} - \frac{1}{N}\sum_l K_{il} - \frac{1}{N}\sum_k K_{kj} + \frac{1}{N^2}\sum_k\sum_l K_{kl}$$

核映射与原映射的对应关系

=> Xc=(X-repmat(Mean,nSmp,1));

Φ(X)' Φ(X)<=>X'X

Kc<=>Xc'Xc

核心代码：

```
W = constructW(fea,options);
%constructKernel[K = constructKernel(fea_a,fea_b,options)]
D = EuDist2(fea_a,[],0);
K = exp(-D/(2*options.t^2));
%constructKernel
D = full(sum(W,2))
[eigvector, eigvalue] = KGE(W, D, options, K);
#KGE[eigvector, eigvalue] = KGE(W, D, options, data)]
K = data;
%映射后的协方差矩阵[Kc]
sumK = sum(K,2);
H = repmat(sumK./nSmp,1,nSmp);
K = K - H - H' + sum(sumK)/(nSmp^2);
K = max(K,K');
[eigvector_PCA, eigvalue_PCA] = eig(K);
eigvalue_PCA = diag(eigvalue_PCA);
K = eigvector_PCA;
DPrime = K*D*K;
DPrime = max(DPrime,DPrime');
WPrime = K*W*K;
WPrime = max(WPrime,WPrime');
dimMatrix = size(WPrime,2);
[eigvector, eigvalue] = eig(WPrime,DPrime);
eigvalue_PCA = eigvalue_PCA.^-1;
eigvector = K*(repmat(eigvalue_PCA,1,length(eigvalue)).*eigvector);
tmpNorm = sqrt(sum((eigvector'*K).*eigvector',2));
eigvector = eigvector./repmat(tmpNorm',size(eigvector,1),1);
#Testing data
 feaTest = rand(5,70);%Test
 Ktest = constructKernel(feaTest,fea,options)
 Y = Ktest*eigvector;
```

Isomap:

Mds,Isomap 通过局部矩阵信息来学习潜在的全局几何特性：MDS 保证局域矩阵的线性距离，而 Isomap 保证测地距离。

Step1:计算确定每个点的领域（根据 Knn 或 ε -neighbors 原则），Dij；

Step2:计算最短路径，先初始化路径距离为 Dg=Dx(i,j)（edge/∞）；再 min{Dg(I,j),Dg(i,k)+Dg(k,j)}，最终 Dg(i,j)即为最短路径.

Step3:计算嵌入坐标 $f^{opt} \models \arg\min_f \sum \left(d_{\mathcal{M}}(\mathbf{x}_i,\mathbf{x}_j) - d(f(\mathbf{x}_i),f(\mathbf{x}_j))\right)^2$ =➔ Specifically, let D be the distance matrix such that D_{ij} is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Define matrix $S_{ij} = D_{ij}^2$ and $H = I - \frac{1}{m}\mathbf{e}\mathbf{e}^T$ where I is the identity matrix and e is the vector of all ones. It can be shown that $\tau(D) = -HSH/2$ is the inner product matrix. That is, $D_{ij}^2 = \tau(D)_{ii} + \tau(D)_{jj} - 2\tau(D)_{ij}, \forall\, i,j$ (Mardia, Kent, & Bibby 1980).

等价于 E=||τ(Dg)-τ(Dy) $||_{L^2}$,其中 τ 表示内积操作； ⬅=====

$||\tau(Dg)-\tau(Dy)||_{L^2} = ||\tau(Dg)-\tau(Dy)||^2$

=tr((τ(Dg)-X'aa'X) (τ(Dg)-X'aa'X)')

= tr(τ(Dg) τ(Dg)')- X'aa'X*τ(Dg)'-τ(Dg)X'aa'X+ X'aa'X X'aa'X

令 a'XX'a=1---->tr(X'aa'X X'aa'X)=tr(a'XX'aa'XX'a)=1,故上式简化为

Argmin : tr(τ(Dg) τ(Dg)')-2tr(a'X τ(Dg)X'a)+1

其中 tr(τ(Dg) τ(Dg)') 与 a 无关，所以上式等价于：

Argmax : a'X τ(Dg)X'a with a'XX'a=1

所以 a'X τ(Dg)X'a = ≺ a'XX'a

考虑到 XX'为奇迹矩阵，令 X=USV'

得 V' τ(Dg)Vb=≺ b;

$a = US^{-1}b;$

其中：τ(Dg)为通过核函数映射到 d(d=2)维平面的协方差矩阵；该除用二项式核。

令$\Phi_i = \Phi(x_i)$，居中$\Phi_i^C = \Phi_i - \frac{1}{N}\sum_k \Phi_k$

$$
\begin{aligned}
K_{ij}^C &= <\Phi_i^C \Phi_j^C> \\
&= (\Phi_i - \frac{1}{N}\sum_k \Phi_k)^T (\Phi_j - \frac{1}{N}\sum_l \Phi_l) \\
&= \Phi_i^T\Phi_j - \frac{1}{N}\sum_l \Phi_i^T\Phi_l - \frac{1}{N}\sum_k \Phi_k^T\Phi_j + \frac{1}{N^2}\sum_k\sum_l \Phi_k^T\Phi_l \\
&= K_{ij} - \frac{1}{N}\sum_l K_{il} - \frac{1}{N}\sum_k K_{kj} + \frac{1}{N^2}\sum_k\sum_l K_{kl}
\end{aligned}
$$

核心代码:

```
#类内间距
for i=1:nLabel
    classIdx = find(options.gnd==Label(i));
    D = EuDist2(data(classIdx,:),[],1);
    G(classIdx,classIdx) = D;
end
#只获取同类间的间隔
for i=1:nLabel
    classIdx = find(options.gnd==Label(i));
    D(classIdx,classIdx) = G(classIdx,classIdx);
end
S = D.^2;
sumS = sum(S);
H = sumS'*ones(1,nSmp)/nSmp;
TauDg = -.5*(S - H - H' + sum(sumS)/(nSmp^2));
TauDg = max(TauDg,TauDg');
sampleMean = mean(data);
data = (data - repmat(sampleMean,nSmp,1));
#LGE 处理argmax aX'WXa=λaX'IXa   [top d]}
[eigvector, eigvalue] = LGE(TauDg, [], options, data);
```

```
    Y = fea*eigvector;
```

LDA：

相对于PCA的无监督学习而言，LDA能将数据的类别考虑进去，实现更好的降维。

假设我们有C个类别，需要K维向量来降维，设这K维向量为a；投影后的结果为Y，Y=a′X；首先，投影前样本的中点为$U_i=\frac{1}{N}\sum_1^N Xi$；投影后的均值为$\widehat{U_l}=\frac{1}{N}\sum_1^N Y=\frac{1}{N}\sum_1^N a'X$=W′ U_i

一方面，我们要使样本中心点尽量分离，即Argmax $S_b=\sum_{k=1}^{c} n_k(\boldsymbol{\mu}^{(k)}-\boldsymbol{\mu})(\boldsymbol{\mu}^{(k)}-\boldsymbol{\mu})^T$,

另一方面，内类的越近越好，即argmin $S_w=\sum_{k=1}^{c}\left(\sum_{i=1}^{n_k}(\mathbf{x}_i^{(k)}-\boldsymbol{\mu}^{(k)})(\mathbf{x}_i^{(k)}-\boldsymbol{\mu}^{(k)})^T\right)$

综上，令$\widehat{S_W}=aSwa$, $\widehat{S_b}=a'Sba$ （第一项为投影后**类内的散列矩阵**之和，第二项为投影后个**各类中心相对于全样本中心的散列矩阵**之和。

最后的J(w)=$\frac{|\widehat{S_B}|}{|\widehat{S_W}|}=\frac{|a'Sba|}{|a'Swa|}$，使之最大化。令 $S_t=\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^T$=XX′= Sb + Sw为**全散列矩阵**

则 $\mathbf{a}_{opt}=\arg\max_{\mathbf{a}}\frac{\mathbf{a}^T S_b\mathbf{a}}{\mathbf{a}^T S_t\mathbf{a}}.$

令均值为0，得 $S_b=\sum_{k=1}^{c} n_k(\boldsymbol{\mu}^{(k)})(\boldsymbol{\mu}^{(k)})^T$

$$=\sum_{k=1}^{c} n_k\left(\frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{x}_i^{(k)}\right)\left(\frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{x}_i^{(k)}\right)^T$$

$$=\sum_{k=1}^{c} X^{(k)}W^{(k)}(X^{(k)})^T$$

$$W_{LDA}=\begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(c)} \end{bmatrix}$$

因而 $S_b=\sum_{k=1}^{c} X^{(k)}W^{(k)}(X^{(k)})^T=XW_{LDA}X^T.$

最后得到 $\mathbf{a}_{opt}=\arg\max_{\mathbf{a}}\frac{\mathbf{a}^T S_b\mathbf{a}}{\mathbf{a}^T S_t\mathbf{a}}=\arg\max_{\mathbf{a}}\frac{\mathbf{a}^T XW_{LDA}X^T\mathbf{a}}{\mathbf{a}^T XX^T\mathbf{a}}.$ 约束a′XX′a=1，即

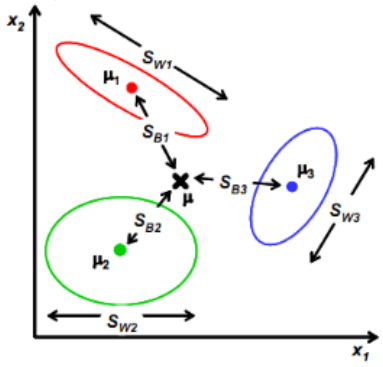
Argmax a’ XWXa= λ a’ XX’ a ；令X=USV′，得

$aUSV'WVSU'a=\lambda aUSV'VSU'a;$

$VWV'b=\lambda b;$

$a=US^{-1}b$

核心代码：




```
#Sdv->US^-1
[U, S, V] = mySVD(data);
[U, S, V]=CutonRatio(U,S,V,options);
eigvalue_PCA = full(diag(S));
data = U;
eigvector_PCA=V*spdiags(eigvalue_PCA.^-1,0,length(eigvalue_PCA),lengt
h(eigvalue_PCA));
#V′WV=Hb′Hb(Suspective)->b
for i = 1:nClass
    index = find(gnd==classLabel(i));
    classMean = mean(data(index,:),1);
    Hb (i,:) = sqrt(length(index))*classMean;
 End
 [dumpVec,eigvalue,eigvector] = svd(Hb,'econ');
eigvector = eigvector_PCA*eigvector;
#确保满秩
if idx < length(eigvalue_PCA)
          U = U(:,1:idx);
          V = V(:,1:idx);
          S = S(1:idx,1:idx);
end
```