東北大學
NORTHEASTERN
UNIVERSITY

R 语言结课报告

题　　目：通过流形学习与稀疏正则化进行多标签特征选择

学　　院：理学院

专　　业：应用统计

姓　　名：李海豹

学　　号：2070015

任课教师：侯婉婷

2021 年 05 月 26 日

1、背景及意义

数据的爆炸式增长，使得统计推断结果达到了前所未有的精确程度，但数据的暴涨，一方面带来了优势，另一方面也带了困扰，即数据中存在着大量的冗余信息和低相关度信息。于是，数据降维方法也随之快速发展起来。

数据降维方法包括特征提取与特征选择两种，前者是构造原始变量的线性组合，以较小的信息损失来极大地减少数据量，这使得计算效率大为提高，但是也使得新变量的可解释性变得较差；后者大致可以分为三类：①过滤式（filter），通过考量单个特征对模型的贡献度或与目标变量的相关性程度对每个变量进行排序，选择出其中最重要的一部分；②包裹式（wrapper），他通过设定一个目标函数，通过对目标对变量进行筛选；③嵌入式（embedding）是先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。类似于 Filter 方法，但是是通过训练来确定特征的优劣。他们都是从原始特征中挑选出对模型贡献较大的一些变量，从而排除一些冗余特征，但是大多数特征选择方法很大程度上忽略了特征之间的相互关系，即单个特征可能对模型的贡献不太显著，但当多个特征相互作用的时候，便对模型产生了较为显著的影响。最优特征子集方法虽然综合考量了各种情况，但是方法的复杂度太大，从而导致不能从根本上解决计算成本较高的问题，例如：当数据集中包含p个特征时，那么最有子集方法就要考虑2^p种情况，当特征数量成千上万时，计算开销将是不可估计的。因此，Tibshirani（1996）提出了 LASSO 方法，它通过对目标函数添加一个l_1正则化惩罚函数，将重要程度较小的变量的系数收缩到零，并在进行特征选择的同时完成了参数估计。l_1正则化对于二分类问题非常有效，但对于多多分类问题显得有些力不从心，于是 Feiping Nie 等（2010）提出$l_{2,1}$正则化来进行多分类问题的特征选择问题，$l_{2,1}$正则化总是可以使得行稀疏，并且对异常点具有鲁棒性，因此也得到了广泛的应用。

以上哪种方法在考虑分类问题时，都是基于欧式距离来判别一个实例与各个类别之间的相似性的。实际上，在维度较高时，会发生维数灾难，即在高维球上，几乎所有样本之间欧式距离都不会相差太大。另一个方面是，欧氏距离并不能度量一些真实的差别，例如，“瑞士卷”数据集上的任意两点，如果考虑其欧氏距

离，那么本来相距较远的两个实例，也可能会在这个标准下被测量的非常接近（见Fig1）。这样的情况在高维空间中可以说是存在的更加普遍，于是，我们不得不换一种方式去考虑实例间的真实差异水平。流形学习（maifold learning 2000）基于数据流形结构重新考量了数据间的差异，流形学习是在高维结构中恢复低维流形，即找到高维流形中的低维流形结构，以实现维数约简和数据可视化。

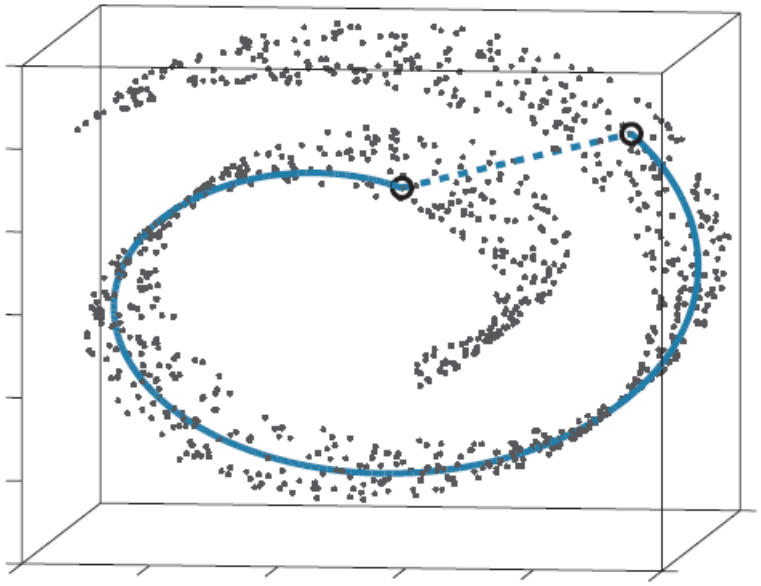

Fig1：虚线展示了两个点之间的欧式距离，实线展示了两点之间的真实距离

随着分类技术的进步与社会需求的提高，单标签分类问题已经不再能够满足需要，很多实例都会同时属于多个类别，例如一本数，可能同时属于政治、历史、哲学等，一条网络信息可能同时出现在微博、知乎、百度、facebook，yahoo 等平台。因此对多分类问题的降维技术的需求也就日益迫切，William Zhu 等（2017）提出基于流形学习与稀疏正则化的方法来进行多标签的特征选择。

2、理论表述

假设数据有m个标签，标签集为$\mathrm{Q}=\{q_1,q_2,\dots,q_m\}$，记$X=[x_1,x_2,\dots,x_n]\in R^{d\times n}$为数据阵，其中$x_i$为第$i$个实例，$y_i=[y_i^1,y_i^2,..,y_i^m]^T$为$x_i$的标签，其中，$y_i^j\in\{0,1\}$，$Y=[y_1,y_2,\dots,y_n]\in R^{m\times n}$，并记$f_i,i=1,2,..,d$为第$i$个特征。

该问题的要优化的损失函数为

$$\sum_{i=1}^{n}||W^Tx_i+b-y_i||_2^2$$

其中$W\in R^{d\times m}$为系数矩阵，$b\in R^m$为偏差。$W=[w_1;w_2;\dots;w_d]$，w_i为W的第i行，他可以视为第i个特征的重要性。

将上述问题表述为矩阵形式，并加以正则化

$$\min_{W,b}\frac{1}{2}||W^TX+b1_n^T-Y||_F^2+\frac{\gamma}{2}R(W)$$

为了从高维特征中采样提取出低维流形特征，需要度量特征之间的相近程度，从上述假设可知，若特征f_i与f_j相近，则其权重向量w_i，w_j也应该相近。所以可以构造流形正则化，将其加入到目标到函数中，流形正则化可以表述为

$$\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}||w_i-w_j||_2^2A_{ij}=Tr(W^T(P-A)W)=Tr(W^TLW)$$

其中P为以对角矩阵，其第 i 对角元为$P_{ii}=\sum_{j=1}^{d}A_{ij}$，$L=P-A$称为 Laplacian 特征图，该方法的关键是特征图的构建，作者在文中也给出了几种构建特征图的方法，

(1)布尔值，特征间的相互依赖关系矩阵由布尔值构成，其中元素A_{ij}通过以下方式确定

$$A_{ij}=\begin{cases}1, & if\ f_j\in N_k(f_i)or\ f_i\in N_k(f_j)\\ 0, & otherwise\end{cases}$$

(2)核方法

$$A_{ij}=\begin{cases}e^{-\frac{||f_i-f_j||_2^2}{t}}, & if\ f_j\in N_k(f_i)or\ f_i\in N_k(f_j)\\ 0, & otherwise\end{cases}$$

其中$t\in R$。

(3)余弦函数

$$A_{ij} = \begin{cases} \dfrac{f_i^T f_j}{||f_i||_2 \cdot ||f_j||_2}, & if\ f_j \in N_k(f_i) or\ f_i \in N_k(f_j) \\ 0, & otherwise \end{cases}$$

上面各式中的 k 是近邻数量的控制参数。

通过流形正则化，可以构造如下目标函数

$$\min_{W,b} \frac{1}{2}||W^T X + b1_n^T - Y||_F^2 + \frac{\alpha}{2} Tr(W^T L W) + \frac{\gamma}{2} R(W)$$

α是流形正则化调整参数。用$||w_i||_2$来衡量特征f_i的重要性，其值越大，该特征就越重要。前面提到降维的思想是使得尽可能多的特征的系数为 0，并使得其对异常点具有鲁棒性，因此我们把正则化项$R(W)$确定为$l_{2,1}$正则化。于是就得到了最终的目标函数

$$\min_{W,b} \frac{1}{2}||W^T X + b1_n^T - Y||_F^2 + \frac{\alpha}{2} Tr(W^T L W) + \frac{\gamma}{2} ||W||_{2,1}$$

其中$||W||_{2,1} = \sum_{i=1}^{d} ||w_i||_2$。

3、优化与算法

下面给出该问题的解

当$w_i \neq 0$时，

$$\frac{\partial ||W||_{2,1}}{\partial \mathrm{W}} = 2\mathrm{DW}$$

其中 D 为一对角矩阵，其第 i 对角元为

$$D_{ii} = \frac{1}{2||w_i||_2}$$

因此，$||W||_{2,1} = Tr(W^T DW)$，此时问题转化为在给定 D 时求解 W，然后在根据 W 更新 D 的迭代过程。

对于任意矩阵A的 Frobenius 矩阵范数，成立

$$||A||_F^2 = \mathrm{Tr}(A^T \mathrm{A})$$

从而目标函数可以重写为

$$g = \frac{1}{2}Tr(Y^T Y) - Tr(W^T XY^T) + \frac{1}{2}Tr(W^T(XX^T + \alpha L + \gamma D)W) + Tr(W^T X 1_n b^T)$$

$$-Tr(Y 1_n b^T) + \frac{1}{2}Tr(b 1_n^T 1_n b^T)$$

g对 b 求偏导，并令其等于 0，有

$$\frac{\partial g}{\partial b} = 0 \Rightarrow b = \frac{1}{n}(Y 1_n - W^T X 1_n)$$

再对 W 求偏导，令其等于 0，有

$$\frac{\partial g}{\partial W} = -XY^T + X 1_n b^T + (XX^T + \alpha L + \gamma D)W = 0$$

将 b 的解带入上式即有

$$\mathrm{W} = (D^{-1} X H_n X^T + \alpha D^{-1} L + \gamma I_d)^{-1} D^{-1} X H_n Y^T$$

其中I_d为 d 阶单位阵，$H_n = I_n - \frac{1}{n}1_n 1_n^T$为中心矩阵（满足对称性和幂等）。

求解 W 的算法可以归结如下：

$Algorithm\ of\ MSSL$

输入：原始数据矩阵X，标签矩阵Y，最终保留的特征数量l，参数α和γ。

输出：筛选出的特征集合I。

1：初始化$t = 0, D^t = I_d$。

2：计算$Laplacian$特征图L。

3：计算中心矩阵$H_n = I_n - \frac{1}{n}1_n 1_n^T$

4：$Repeat$

5：　　计算

$$W^{t+1} = ((D^t)^{-1}XH_nX^T + \alpha(D^t)^{-1}L + \gamma I_d)^{-1}(D^t)^{-1}XH_nY^T$$

6：　　计算对角矩阵D^{t+1},其第i对角元为$D_{ii}^{t+1} = \frac{1}{2|w_i^{t+1}|||_2}$

7：　　$t = t + 1$

8：$Until$满足收敛条件

9：对$\left|\left|w_i^{t+1}\right|\right|_2$进行降序排列，并取前$l$个元素对应的指标构成最终选择的特征子集。

4、实例分析

选取来自多标签学习网站(http://mulan.sourceforge.net/datasets-mlc.html)的 6 个数据集，数据集相关信息如下

Dataset	$Dim(D)$	$L(D)$	*Training Set*			*Test Set*		
			$\|D\|$	$PMC(\%)$	ANL	$\|D\|$	$PMC(\%)$	ANL
Computer	681	33	2000	29.60	1.487	3000	31.27	1.522
Reference	793	33	2000	13.75	1.159	3000	14.60	1.177
Business	438	30	2000	42.20	1.590	3000	41.93	1.586
Health	612	32	2000	42.20	1.667	3000	47.20	1.659
Scene	294	6	1211	6.11	1.062	1196	8.61	1.086
Yeast	103	14	1500	98.40	4.228	917	99.13	4.252

Table1：各实验数据集及其相关描述

其中 Dim(D)是数据集的特征数量，L(D)是数据集的标签数量，$|D|$是数据集的实例数，$PMC(\%)$是同时属于多个标签的实例占总实例的比例，ANL是平均每个实例同时属于的标签数量。

下面选择了几种较为常用且有效的特征选择算法进行比较（Baseline 使用了全部特征），以平均精准度为评估标准，其结果如下：

Algorithms	MSSL	RFS	MDMR	PMU	FIMF	LASSO	Baseline
Computer	**0.6510**	0.6420	0.6305	0.6092	0.6203	0.6315	0.6334
Reference	**0.6547**	0.6370	0.6493	0.6034	0.6477	0.6414	0.6193
Business	**0.8806**	0.8723	0.8757	0.8691	0.8730	0.8745	0.8798
Health	**0.7413**	0.7328	0.7257	0.6594	0.7089	0.7382	0.6812
Scene	0.8287	0.8278	0.7633	0.8034	0.6906	0.8312	**0.8512**
Yeast	**0.7674**	0.7598	0.7580	0.7563	0.7552	0.7566	0.7585

Table2：各种方法在不同数据集中的平均精准度表现

表现最优的结果以加粗字体显示，次优的结果加下划线显示。可以看到，几乎在所有的数据集中，MSSL 方法的表现都显著优于其他方法。

各种方法在不同数据集下的 hamming loss 如下表所示

Algorithms	MSSL	RFS	MDMR	PMU	FIMF	LASSO	Baseline
Computer	**0.0380**	0.0390	0.0398	0.0416	0.0409	0.0397	0.0412
Reference	**0.0272**	0.0282	0.0279	0.0301	0.0284	0.0287	0.0314
Business	**0.0264**	0.0274	0.0273	0.0284	0.0274	0.0276	0.0269
Health	**0.0370**	0.0384	0.0391	0.0457	0.0407	0.0381	0.0458
Scene	0.1047	0.1074	0.1348	0.1137	0.1587	0.1066	**0.0989**
Yeast	**0.1940**	0.1963	0.1999	0.2006	0.2021	0.2000	0.1980

Table3：各种算法在不同数据集中下的 Hamming Loss

同样地，MSSL 方法再一次表现出了其优秀的性能，其 hamming loss 几乎在所有数据集下都是最小的。

下面的图展示了各种方法在不同数据集上选择保留不同的特征数量时，其平均精准率的变化曲线

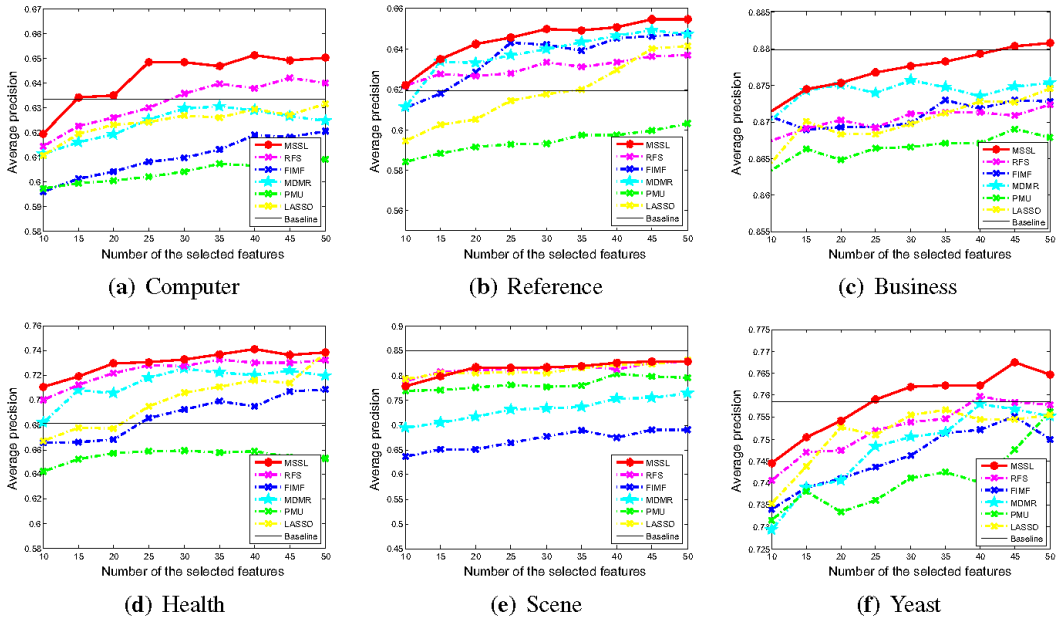

Fig2：X 轴为保留的特征数量，Y 轴为平均准确率，水平实线为 Baseline 方法的平均准确率。

再一次，MSSL 方法展示了在保留不同特征数目时，其平均准确率都要优于其他方法，而数据集 Scene 依然是一个特例，我们所选择的任何特征选择方法都不能优于 Baseline 的水平。

关于在不同数据集上不同方法上保留不同特征时的 hamming loss 变化情况如下图所示

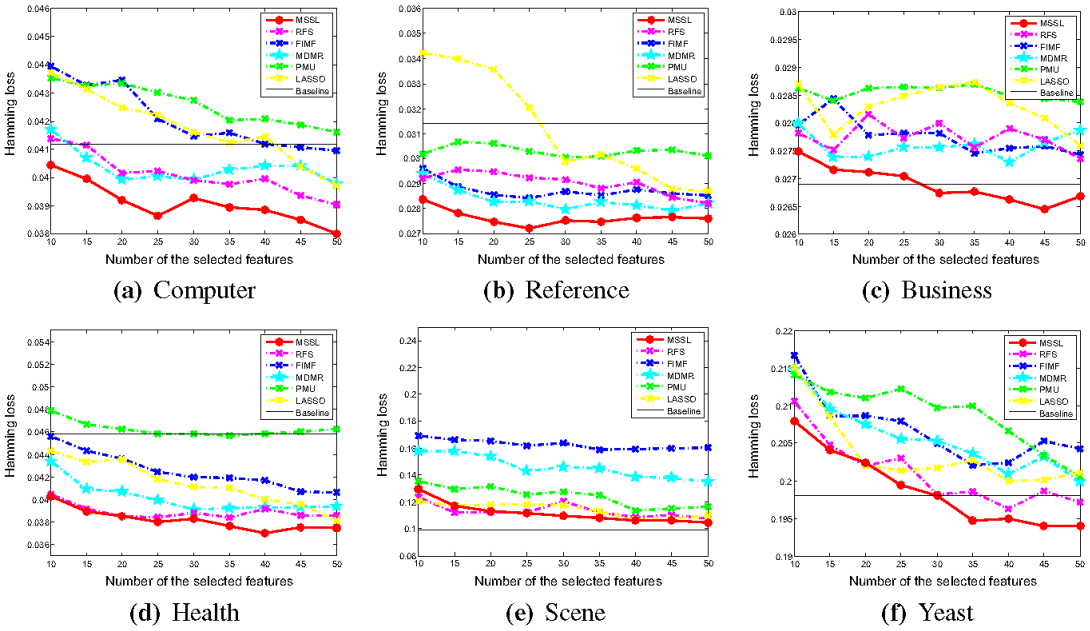

Fig3：X 轴为保留的特征数量，Y 轴为 hamming loss，hamming loss 越小越好

以上不同方法在不同数据集上的表现，可以发现 MSSL 方法的表现几乎一致优于其他方法，这也说明了在高维数据中提取其低维的流形结构是至关重要的，它能够使得对数据的判断更加接近实际。

接下来，通过设定不同的 k 值来讨论 MSSL 方法随该参数的变化，以确定 MSSL 是否对其敏感。

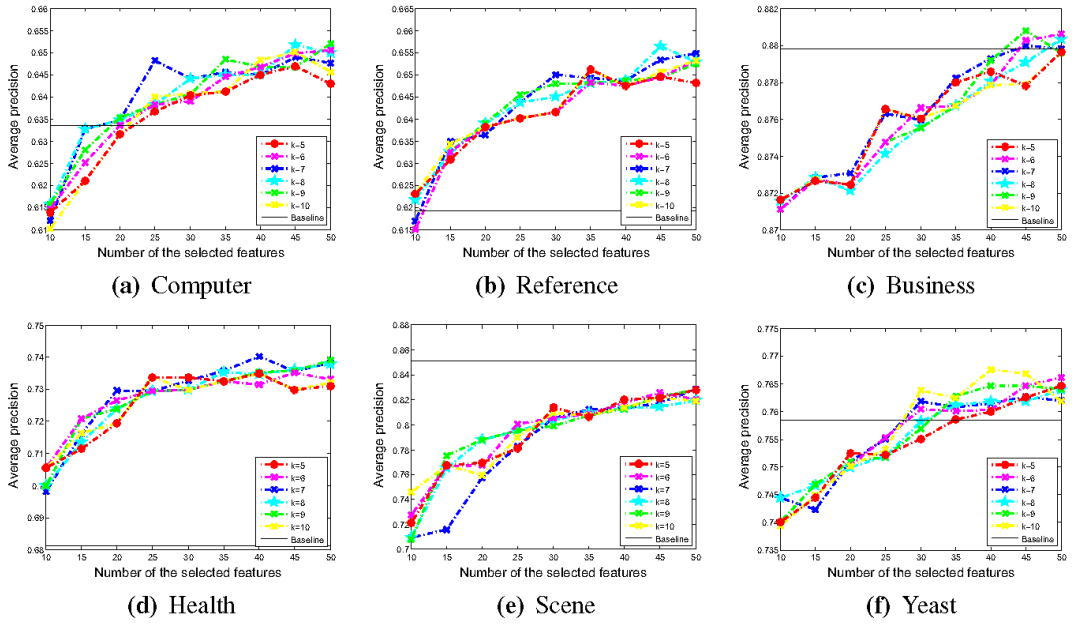

Fig4：X 轴为保留的特征数量，Y 轴为平均准确率

从图中可以看得出，对于不同的 k 值，各条曲线几乎粘连在一起，并无明显差别，所以该方法对参数 k 的选取式也是不敏感的。

5、个人观点及设想

本文有一个不足之处，即并未指明算法的收敛条件。如果以单特征权重的值作为收敛准则，那么可能会错过其他应该被舍弃的特征；如果以固定的迭代次数为标准，就需要提前测试其收敛的迭代次数，这会浪费很多时间，从而导致低效率。因此考虑以正则化的范数值作为收敛准则，正则化项本身就是一个约束，作为判敛条件也非常合理。另一个考虑是，当冗余特征的权重收缩为 0 之后，其余特征的值在每次迭代中的变化都不会太大，所以，考虑权重矩阵 W 的 Frobenius 矩阵范数的变化的比例作为考量指标，当这个比例小于以给定常数时，即可认为收敛。

流形学习确能够有效提取高维空间中的低维流形，从而使得数据在降维的过程中保持了其原有的特质，进一步保留了数据的有效信息。流形学习考虑了特征之间的相互关系，即更全面的使用了数据包含的有用信息，因此将使得后续推断更加精准有效。

但目前通过流形学习进行多标签数据的特征选择还仅限于监督学习，而作者也将半监督学习作为下一步的工作方向。因此，将流形学习放诸无监督学习是一个有研究价值的方向。Xiaofeng Zhu(2018)提出将 self-space learning regulation 应用到无监督特征选择中，他的思想是值得借鉴的，但要指出，Zhu 在文中主要考虑的是异常点的处理。

6、参考文献

[1] Cai, Zhiling, Zhu, et al. Multi-label feature selection via feature manifold learning and sparsity regularization[J]. International journal of machine learning and cybernetics, 2018.

[2] Tibshirani R J . Regression Shrinkage and Selection via the LASSO[J]. Journal of the Royal Statistical Society. Series B: Methodological, 1996, 73(1):273-282.

[3] Zheng W , Zhu X , Wen G , et al. Unsupervised feature selection by self-paced learning regularization[J]. Pattern Recognition Letters, 2018:S0167865518302782-.

[4] Tenenbaum J B , Silva V D , Langford J C . A Global Geometric Framework for Nonlinear Dimensionality Reduction[J]. Science, 2001, 290(5500):2319-2323.

[5] Nie F , Huang H , Xiao C , et al. Efficient and Robust Feature Selection via Joint ℓ2, 1-Norms Minimization[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2010.

7、算法的 R 实现

（以 scene 数据集为例）

predefined functions

“Inv.R”

```
inv <- function(x){
   "
   param x: matrix
                  x must be a square matrix
    return the inverse matrix of x
   "
   n <- dim(x)[1]
   I <- diag(n)
   return(solve(x,I))
}
```

“GraphLaplacian.R”

```
library("philentropy")    # load this library to solve large datasets’ distances matrix
GraphLaplacian <- function(x, k = 7, method = NULL){
   "
   param x: matrix
            data matrix was composed of all of features
   param k: int, options
            the number of neighbours in knn method
            default is 7
   param method: str, options
            given the method to solve Graph Laplacian
            default is NULL; in this case, method of heat kernel would be used
```

```
                choose in 'boolean', 'kernel', 'cosine'
return L: matrix
                the Graph Laplacian
"

n <- dim(x)[1]
d <- dim(x)[2]

if(is.null(method)){
   method <- "kernel"
}

A <- matrix(data = 0, nrow = d, ncol = d)
P <- diag(d)
distances <- distance(t(x), method = "euclidean") + diag(d)*10000

if (method=="boolean"){
   for (i in c(1:d)){
      top.k <- sort(distances[i,], decreasing = FALSE)[k]
      A[i,] <- (distances[i,] < top.k)
      P[i,i] <- sum(A[i,])
   }
}

if (method=="kernel"){
   sigma <- 1
   for (i in c(1:d)){
      top.k <- sort(distances[i,], decreasing = FALSE)[k]
      position <- (distances[i,]>top.k)
```

```
            A[i,] <- exp(-distances[i,]/(2*sigma))
            A[i,][position] <- 0
            P[i,i] <- sum(A[i,])
        }
    }

    if (method=="cosine"){
        for (i in c(1:d)){
            top.k <- sort(distances[i,], decreasing = FALSE)[k]
            position <- (distances[i,]>top.k)
            for (j in c(i:d)){
                A[i,j] <- (t(x[,i])%*%x[,j])/(sqrt(sum(x[,i]^2))*sqrt(sum(x[,j]^2)))
                A[j,i] <- A[i,j]
            }
            A[i,][position] <- 0
            P[i,i] <- sum(A[i,])
        }
    }

    L <- P - A

    return(L)
}
```

“L21.R”

```
L21 <- function(w){
    "
    param w: matrix
```

```
            Matrix of weights
    return s: float
            value of L2,1 regularization
    "

    d <- dim(w)[1]
    temp <- matrix(data = 0, nrow = d, ncol = 1)

    for (i in c(1:d)){
        temp[i] <- sqrt(sum(w[i,]^2))
        s <- sum(temp)
    }

    return(s)
}
```

“SolveW.R”

```
SolveW <- function(x, y, alpha = 1, gamma = 10, maxIte = 50){
    "
    param x: matrix
            the data matrix is composed of all of features
    param y: matrix
            the label matrix
    param alpha: float
            coefficient of manifold regularization
    param gamma: float
            coefficient of L2,1regularization
    param maxIte: int
            the maximum of iteration
```

```
    return W: matrix
                    matrix of weights
    "

    n <- dim(x)[1]
    d <- dim(x)[2]

    L <- GraphLaplacian(x, 7, "boolean")
    I <- diag(n)
    E <- matrix(data = 1, nrow = n, ncol = 1)
    H <- I - E%*%t(E)/n
    v <- E[c(1:d)]
    obj <- matrix(data = 0, nrow = maxIte, ncol = 1)

    for (i in c(1:maxIte)){
        D <- diag(v)
        S <- inv(D%*%((t(x)%*%H%*%x) + alpha*L) +
gamma*diag(d))%*%(D%*%t(x)%*%H%*%y)
        b <- (t(y)%*%E - t(S)%*%t(x)%*%E)/n
        for (j in c(1:d)){
            v1 <- sqrt(sum(S[j,]^2))
            v[j] <- 2*v1
        }
        obj[i] <- 0.5*sum((y - x%*%S-E%*%t(b))^2) +
0.5*alpha*sum(diag(t(S)%*%L%*%S)) + 0.5*gamma*L21(S)
    }

    return(S)
}
```

“FeatureSelection.R”

```
FeatureSelection <- function(W, l = 45){
    "
    param W: matrix
                matrix of weights
    param l: int
                the number of preserved features
    return I: vector
                the set of index of preserved features
    "

    d <- dim(W)[1]
    weights <- matrix(data = 0, nrow = 1, ncol = l)

    for (i in c(1:d)){
        weights[i] <- sqrt(sum(W[i,]^2))
    }

    I <- sort(order(weights)[c(1:l)], decreasing = FALSE)

    return(I)
}

#### main function ####
```

“MSSL.R”

```
# load all of predefined functions
dir()
source("inv.R")
```

```
source("GraphLaplacian.R")
source("L21.R")
source("SolveW.R")
source("FeatureSelection.R")

# the path of data file
path.train <- " scene-train.csv"
path.test <- " scene-test.csv"

# 294 features and 6 labels were contained in this datasets
data.train <- as.matrix(read.csv(path.train, header = TRUE))
data.test <- as.matrix(read.csv(path.test, header = TRUE))
x.train <- data.train[, c(1:294)]
x.test <- data.test[, c(1:294)]
y.train <- data.train[, c(295:300)]
Y <- data.test[, c(295:300)]

W <- SolveW(x.train, y.train, 1, 10, 50)
I <- FeatureSelection(W)    # index of preserved features
W <- W[I,]    # preserved features
X <- x.test[,I]
n <- dim(X)[1]
d <- dim(X)[2]
m <- dim(Y)[2]
E <- matrix(data = 1, nrow = n, ncol = 1)    # a column vector fill with elements 1
b <- (t(Y)%*%E - t(W)%*%t(X)%*%E)/n
Y.pre <- X%*%W + E%*%t(b)
Y.label <- (Y.pre>0.5)
Y.label[Y.label==TRUE] <- 1    # logic values transform to numeric values
```

```
Y.label[Y.label==FALSE] <- 0
AveragePrecision <- sum((Y+Y.label)==2)/sum(Y.label)
HammingLoss <- sum(xor(Y,Y.label))/(n*d)
```