图Lasso求逆协方差矩阵(Graphical Lasso for inverse covariance matrix)

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

1．图Lasso方法的基本理论

图 Lasso 是一种可以快速估计逆协方差矩阵的算法，它使用 l_1 罚来增加逆协方差矩阵的稀疏性，并使用快速坐标下降法来解决单个 Lasso 问题，当数据的维数较高时计算速度也很快。

2.1.1 图 Lasso 方法的罚图模型

假设数据服从多元高斯分布，估计数据的无向图模型则相当于估计它的逆协方差矩阵。对于数据的无向图模型，一个节点代表一个特征，两个节点之间的关系用边来表示。对于高斯分布的数据，其逆协方差矩阵 Σ^{-1} 中的元素表示边是否存在，逆协方差矩阵中的非零元素表示一对节点之间有边存在，而一对变量条件独立对应于逆协方差矩阵中的零元素，即如果 Σ^{-1} 的第$(i,\ j)$个元素是 0，则在给定其他变量的情况下，第 i 个变量和第 j 个变量是条件独立的。

假设我们有 n 个相互独立且服从高斯分布的样本，每个样本特征均为 p 维，均值为 μ，协方差矩阵为 Σ。传统的估计 Σ^{-1} 的方法是通过最大化数据的对数似然函数，令 $\mathrm{S}=\frac{\mathrm{X}^T\mathrm{X}}{n}$ 表示数据的协方差矩阵，在高斯模型中，对数似然函数的形式为

$$\log\det\Sigma^{-1}-tr\left(\mathrm{S}\Sigma^{-1}\right) \tag{2-1}$$

式中　det——行列式；

　　tr——迹；

令 $\boldsymbol{\Theta}=\boldsymbol{\Sigma}^{-1}$，关于 $\boldsymbol{\Theta}$ 最大化(2-1)式将产生最大似然估计 $\hat{\boldsymbol{\Theta}}=\mathrm{S}^{-1}$，但是使用这种最大似然方法来估计 $\boldsymbol{\Sigma}^{-1}$ 存在一些问题。当特征维数大于样本数目时，S 是奇异矩阵，数据的最大似然估计将无法计算。当特征维数近似等于样本数目时，即使 S 不是奇异矩阵，数据的最大似然估计也将面临很复杂的计算。另外，这种最大似然方法通常不会得到准确为零的元素，得到的图模型中特征对之间的条件独立关系将会很复杂。

在文献[8]中，Yuan 和 Lin 等人提出了一个罚对数似然函数代替原先的似然函数，即对整个逆协方差矩阵 $\boldsymbol{\Theta}$ 取 l_1 罚，数据的罚对数似然函数为

$$\log\det\boldsymbol{\Theta}-tr\left(\mathrm{S}\boldsymbol{\Theta}\right)-\rho\left\|\boldsymbol{\Theta}\right\|_1 \tag{2-2}$$

式中　$\boldsymbol{\Theta}$——逆协方差矩阵；

　　ρ——罚参数，$\rho>0$；

关于$\boldsymbol{\Theta}$最大化罚对数似然函数(2-2)式可估计出数据的罚图模型，通过这种最大化罚对数似然函数的方法来估计$\boldsymbol{\Sigma}^{-1}$能克服原先的最大似然方法存在的缺点：当特征维数大于样本数目时，关于$\boldsymbol{\Theta}$最大化(2-2)式也能求解；当罚参数取值很大时，估计的$\boldsymbol{\Theta}$能得到准确为零的元素，即罚参数能控制$\boldsymbol{\Theta}$的稀疏程度。

2.1.2 图 Lasso 方法的基本思想

图 Lasso 方法由 Friedman 等人提出，使用该方法估计稀疏无向图模型，其中以块坐标下降算法为出发点，并通过快速坐标下降法来解决图 Lasso 方法中 Lasso 问题。

图 Lasso 方法用来求解最大化罚对数似然函数问题，即图 Lasso 问题的目标函数是最大化(2-2)式，如下式

$$\hat{\boldsymbol{\Theta}} = \max \log\det \boldsymbol{\Theta} - tr\left(\mathrm{S}\boldsymbol{\Theta}\right) - \rho\left\|\boldsymbol{\Theta}\right\|_1 \tag{2-3}$$

令 W 为$\boldsymbol{\Sigma}$的估计，将 W 分成四块，分别为W_{11}，$\boldsymbol{\omega}_{12}$，$\boldsymbol{\omega}_{12}^{\mathrm{T}}$和$\omega_{22}$，具体按以下形式分块

$$\mathrm{W} = \begin{pmatrix} \mathrm{W}_{11} & \boldsymbol{\omega}_{12} \\ \boldsymbol{\omega}_{12}^{\mathrm{T}} & \omega_{22} \end{pmatrix} \tag{2-4}$$

其中，W_{11}是一个$p-1\times p-1$维的矩阵，$\boldsymbol{\omega}_{12}$是一个$p-1$维的列向量，ω_{22}是一个标量。假设在一个p维问题中，则共有p种不同的 W 分块形式。对$i=1,2,...,p$，第i种 W 的分块中左上角块W_{11}由除去 W 第i行和第i列的矩阵组成，右上角块$\boldsymbol{\omega}_{12}$由 W 的第i列除去第i项的向量组成，右下角块ω_{22}则为 W 的第i行的第i项，以此类推。由于协方差矩阵的对角线元素均大于0，则 W 的对角线元素均大于0，包括W_{11}中的全部对角线元素和ω_{22}。用ω_{ii}表示 W 的对角线元素，则对所有的i都有$\omega_{ii}>0$。

将 S 按 W 的分块规律分成四块，分别为S_{11}，s_{12}，$\mathrm{s}_{12}^{\mathrm{T}}$和$s_{22}$，具体按以下形式分块

$$\mathrm{S} = \begin{pmatrix} \mathrm{S}_{11} & \mathrm{s}_{12} \\ \mathrm{s}_{12}^{T} & s_{22} \end{pmatrix} \tag{2-5}$$

其中，S_{11}是$p-1\times p-1$维的矩阵，s_{12}是$p-1$维的列向量，s_{22}是一个标量。

将$\boldsymbol{\Theta}$按 W 的分块规律分成四块，分别为$\boldsymbol{\Theta}_{11}$，$\boldsymbol{\theta}_{12}$，$\boldsymbol{\theta}_{12}^{\mathrm{T}}$和$\theta_{22}$，具体按以下形式分块

$$\boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{\Theta}_{11} & \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}^{T} & \theta \end{pmatrix} \tag{2-6}$$

示逆协方差矩阵$\boldsymbol{\Theta}$的对角线元素，由于W的对角线元素$\omega_{ii}>0$，则对所有的i都有$\theta_{ii}>0$。

协方差矩阵W的右上角块$\boldsymbol{\omega}_{12}$满足下式

$$\hat{\omega}_{12}=\arg\min_{\omega_{12}}\left\{\boldsymbol{\omega}_{12}{}^{\mathrm{T}}\mathrm{W}_{11}^{-1}\boldsymbol{\omega}_{12}:\left\|\boldsymbol{\omega}_{12}-\mathrm{s}_{12}\right\|_{\infty}\le\rho\right\} \tag{2-7}$$

(2-7)式是一个箱约束二次规划问题，可以使用内点法解决，通过交换行和列，每一列相当于解决一个(2-7)式的问题，每一个阶段后更新对W的估计，一直重复更新，直到W收敛。

利用凸对偶性，(2-7)式相当于解决以下对偶问题

$$\min_{\boldsymbol{\beta}}\left\{\frac{1}{2}\left\|\mathrm{W}_{11}^{1/2}\boldsymbol{\beta}-\mathrm{W}_{11}^{-1/2}\mathrm{s}_{12}\right\|^{2}+\rho\left\|\boldsymbol{\beta}\right\|_{1}\right\} \tag{2-8}$$

其中满足$\boldsymbol{\omega}_{12}=\mathrm{W}_{11}\boldsymbol{\beta}$，$\boldsymbol{\beta}$可以解决(2-8)式，通过$\boldsymbol{\omega}_{12}=\mathrm{W}_{11}\boldsymbol{\beta}$可以解决(2-7)式。$\hat{\boldsymbol{\beta}}$是一个稀疏的向量，因此计算$\boldsymbol{\omega}_{12}=\mathrm{W}_{11}\hat{\boldsymbol{\beta}}$时速度会很快。

实际上，(2-8)式与图Lasso的目标函数(2-3)式是等价的，首先我们可以直接证明(2-3)式和(2-8)式相等。由$\mathrm{W}=\boldsymbol{\Sigma}$和$\boldsymbol{\Theta}=\boldsymbol{\Sigma}^{-1}$可得$\mathrm{W}\boldsymbol{\Theta}=\mathrm{I}$，则可得到以下表达式：

$$\begin{pmatrix}\mathrm{W}_{11} & \boldsymbol{\omega}_{12}\\ \boldsymbol{\omega}_{12}^{\mathrm{T}} & \omega_{22}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Theta}_{11} & \boldsymbol{\theta}_{12}\\ \boldsymbol{\theta}_{12}^{\mathrm{T}} & \theta_{22}\end{pmatrix}=\begin{pmatrix}\mathrm{I} & 0\\ 0^{\mathrm{T}} & 1\end{pmatrix} \tag{2-9}$$

由(2-9)式可得到

$$\mathrm{W}_{11}\boldsymbol{\theta}_{12}+\boldsymbol{\omega}_{12}\theta_{22}=0 \tag{2-10}$$

$$\boldsymbol{\omega}_{12}^{\mathrm{T}}\boldsymbol{\theta}_{12}+\omega_{22}\theta_{22}=1 \tag{2-11}$$

对(2-3)式求次梯度的等式为

$$\mathrm{W}-\mathrm{S}-\rho\cdot\boldsymbol{\Gamma}=0 \tag{2-12}$$

其中$\log\det\boldsymbol{\Theta}$对$\boldsymbol{\Theta}$的导数为$\boldsymbol{\Theta}^{-1}=\mathrm{W}$，$tr\left(\mathrm{S}\boldsymbol{\Theta}\right)$对$\boldsymbol{\Theta}$的导数为S，$\left|\boldsymbol{\Theta}\right|$的次梯度为$\boldsymbol{\Gamma}$。$\boldsymbol{\Gamma}$是一个$p\times p$维矩阵，这里$\Gamma_{ij}\in sign\left(\Theta_{ij}\right)$，其中$\Gamma_{ij}$指的是$\boldsymbol{\Gamma}$第$i$行第$j$列元素，$\Theta_{ij}$指的是$\boldsymbol{\Theta}$第$i$行第$j$列元素，$sign$是符号函数。如果$\Theta_{ij}\neq 0$则$\Gamma_{ij}=sign\left(\Theta_{ij}\right)$，如果$\Theta_{ij}=0$则$\Gamma_{ij}\in\left[-1,1\right]$。

将(11)式按上文所示的矩阵分块的形式展开，如下

$$\begin{pmatrix}\mathrm{W}_{11} & \boldsymbol{\omega}_{12}\\ \boldsymbol{\omega}_{12}^{\mathrm{T}} & \omega_{22}\end{pmatrix}-\begin{pmatrix}\mathrm{S}_{11} & \mathrm{s}_{12}\\ \mathrm{s}_{12}^{\mathrm{T}} & s_{22}\end{pmatrix}-\rho\begin{pmatrix}\boldsymbol{\Gamma}_{11} & \boldsymbol{\gamma}_{12}\\ \boldsymbol{\gamma}_{12}^{\mathrm{T}} & \gamma_{22}\end{pmatrix}=0 \tag{2-13}$$

其中$\boldsymbol{\Gamma}_{11}\in sign\left(\Theta_{11}\right)$是一个$p-1\times p-1$维的矩阵，$\boldsymbol{\gamma}_{12}\in sign\left(\theta_{12}\right)$是$p-1$维的列向量，$\gamma_{22}\in sign\left(\theta_{22}\right)$是一个标量。

$$\omega_{12} - s_{12} - \rho \cdot \gamma_{12} = 0 \tag{2-14}$$

对(2-8)式求出的次梯度等式为

$$W_{11}\beta - s_{12} + \rho \cdot \upsilon = 0 \tag{2-15}$$

其中 $\upsilon \in sign(\beta)$，若 $\beta_i \neq 0$ 则有 $\upsilon_i \in sign(\beta_i)$，若 $\beta_i = 0$ 则有 $\upsilon_i \in [-1,1]$。

由(2-10)式可得出 $\theta_{12} = -W_{11}^{-1}\omega_{12}\theta_{22}$ ，由于满足 $\theta_{22} > 0$ 且 $\beta = W_{11}^{-1}\omega_{12}$，则可得出 $sign(\theta_{12}) = sign(-W_{11}^{-1}\omega_{12}\theta_{22}) = -sign(W_{11}^{-1}\omega_{12}) = -sign(\beta)$，即可得出 $\upsilon = -\gamma_{12}$。由于满足 $\omega_{12} = W_{11}\beta$，$\upsilon = -\gamma_{12}$，即可得(2-14)式和(2-15)式相等，则(2-3)式和(2-8)式相等。因此，目标函数由求解(2-3)式变成求解(2-8)式。

2．坐标下降算法

坐标下降算法经常在求解多变量目标函数的最优解时使用，每次沿一个方向最小化目标函数 ，这样最小化问题可以很快完成。

2.2.1 坐标下降算法求解 Lasso 问题

Lasso 方法由 Tibshirani 提出，该方法通过最小化残差的平方和且约束条件为系数的绝对值之和小于某个特定常数来估计参数[36]，Friedman 于 2006 年提出使用坐标下降算法来求解 Lasso 问题。

假设每个数据样本用一个 P 维特征向量来描述 P 个特征的值，即 $\mathrm{X}=\{\mathrm{X}_1,\mathrm{X}_2,\ldots,\mathrm{X}_j,\ldots\mathrm{X}_P\}$，样本类别用向量 Y 表示，样本数目用 N 表示，则有 $\mathrm{X}_j=\{x_{1j},x_{2j},\ldots,x_{Nj}\}^T$ 与 $\mathrm{Y}=\{y_1,y_2,\ldots,y_N\}^T$，Lasso 问题可描述为

$$\min_{\beta}\frac{1}{2}\left\|\mathrm{y}-\mathrm{X}\beta\right\|_2^2+\rho\left\|\beta\right\|_1 \tag{2-16}$$

对(2-16)式求次梯度等式可得

$$\mathrm{X}^{\mathrm{T}}\mathrm{X}\beta-\mathrm{X}^{\mathrm{T}}\mathrm{y}+\rho\cdot sign(\beta)=0 \tag{2-17}$$

下面是 Lasso 回归问题通过坐标下降法求解时 β 的详细步骤，更新为如下形式所示

$$\hat{\beta}_j=\begin{cases}\dfrac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right)-\rho & \dfrac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right)>0\text{且}\left|\dfrac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right)\right|>\rho\\ \dfrac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right)+\rho & \dfrac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right)<0\text{且}\left|\dfrac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right)\right|>\rho\\ 0 & \left|\dfrac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right)\right|<\rho\end{cases} \tag{2-18}$$

对于 $j=1,2,...,p,1,2,...,p,...$ ，(2-18)式的更新一直重复，直到收敛。这个更新可以写成软阈值形式

$$\beta_j(\gamma)=S\left(\frac{1}{N}\sum_{i=1}^{N}x_{ij}\left(y_i-\tilde{y}_i^{(j)}\right),\gamma\right) \tag{2-19}$$

其中 $j=1,2,...,p,1,2,...,p,...$ ，上式为 Lasso 回归问题参数估计的迭代更新公式。S 是软阈值操作符，满足

$$S(x,t)=\mathrm{s}ign(x)(|x|-t)_+ \tag{2-20}$$

2.2.2 坐标下降算法求解图 Lasso 问题

(2-8)式的次梯度等式(2-15)式可写为 $\mathrm{W}_{11}\beta-\mathrm{s}_{12}+\rho\cdot sign(\beta)=0$，则 W_{11} 相当于(2-17)式中的 $\mathrm{X}^{\mathrm{T}}\mathrm{X}$，$\mathrm{s}_{12}$ 相当于(2-17)式中的 $\mathrm{X}^{\mathrm{T}}\mathrm{y}$，所以(2-8)式类似于一个 Lasso 回归问题，可以由坐标下降法有效地解决。下面是求解 β 的详细步骤：令 $\mathrm{V}=\mathrm{W}_{11}$，$\mathrm{u}=\mathrm{s}_{12}$，更新由如下形式所示

$$\hat{\beta}_j=\begin{cases}u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k-\rho & u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k>0 \text{ 且} \left|u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k\right|>\rho \\ u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k+\rho & u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k<0 \text{ 且} \left|u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k\right|>\rho \\ 0 & \left|u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k\right|<\rho \end{cases} \tag{2-21}$$

对于 $j=1,2,...,p-1,1,2,...,p-1,...$ (2-21)式的更新一直重复，直到收敛。这个更新可以写为软阈值形式

$$\hat{\beta}_j\longleftarrow\frac{S(u_j-\sum_{k\neq j}V_{jk}\hat{\beta}_k,\rho)}{V_{jj}} \tag{2-22}$$

其中 $j=1,2,...,p-1,1,2,...,p-1,...$ ，S 是与(2-20)式相同的软阈值操作符。当 W 变化的平均绝对值小于 $t\cdot ave|\mathrm{S}^{-diag}|$ 时，算法将停止更新，其中 S^{-diag} 是协方差矩阵 S 的非对角元素，t 是一个固定阈值，可根据经验进行选择，本文选取 $t=0.001$。

3．图Lasso算法

2.3.1 图 Lasso 算法估计协方差矩阵

图Lasso问题的目标函数(2-3)式等价于(2-8)式，则由求解(2-3)式变成对(2-8)式的求解。首先，将 W 分成 W_{11}， $\boldsymbol{\omega}_{12}$， $\boldsymbol{\omega}_{12}^{T}$ 和 ω_{22}，将分成 S_{11}， s_{12}， s_{12}^{T} 和 s_{22}，通过坐

标下降算法求解出(2-8)式的 β。一般情况下，$\hat{\boldsymbol{\beta}}$ 是稀疏的，因此计算 $\boldsymbol{\omega}_{12}=\mathrm{W}_{11}\hat{\boldsymbol{\beta}}$ 时速度会很快，若 $\hat{\boldsymbol{\beta}}$ 有 r 个非零元素，则仅进行 rp 次运算。其次，通过 $\boldsymbol{\omega}_{12}=\mathrm{W}_{11}\boldsymbol{\beta}$ 更新一次对 W 的估计。每更新p次 $\boldsymbol{\omega}_{12}$ 则 W 完全更新一次。一直重复更新，直到 W 收敛。

逆协方差矩阵 $\boldsymbol{\Theta}$ 的每一个对角线元素均满足 $\theta_{ii}>0$，在对(2)式求次梯度的等式 $\mathrm{W}-\mathrm{S}-\rho\cdot\boldsymbol{\Gamma}=0$ 中，有 $\Gamma_{ij}\in sign\left(\Theta_{ij}\right)$，即可得 $\Gamma_{ii}=1$，则对所有的 i 均满足 $\omega_{ii}=s_{ii}+\rho$。

2.3.2 图 Lasso 算法估计逆协方差矩阵

估计出协方差矩阵 W 后，我们也很容易求出逆协方差矩阵 $\hat{\boldsymbol{\Theta}}=\mathrm{W}^{-1}$。由(2-10)式和(2-11)式化解可得到

$$\theta_{12}=-\mathrm{W}_{11}^{-1}\omega_{12}\theta_{22} \tag{2-23}$$

$$\theta_{22}=1/\left(\omega_{22}-\omega_{12}^{\mathrm{T}}\mathrm{W}_{11}^{-1}\omega_{12}\right) \tag{2-24}$$

又因为 $\hat{\beta}=\mathrm{W}_{11}^{-1}\omega_{12}$，则

$$\hat{\theta}_{12}=-\hat{\beta}\theta_{22} \tag{2-25}$$

$$\hat{\theta}_{22}=1/\left(\omega_{22}-\omega_{12}^{\mathrm{T}}\hat{\beta}\right) \tag{2-26}$$

在图 Lasso 算法的更新过程中，一次 W 的更新中共有 p 个问题，将每次更新时计算出来的 p 组系数 $\hat{\beta}$ 储存在一个矩阵 $\hat{\mathrm{B}}$ 中。图 Lasso 算法在收敛后即可利用最后得到的矩阵 $\hat{\mathrm{B}}$ 直接计算(2-25)式和(2-26)式，从而可以快速求出逆协方差矩阵 $\hat{\boldsymbol{\Theta}}$。

2.3.3 图 Lasso 算法的步骤

下面的步骤中 W 的对角线元素保持不变，图Lasso算法的具体步骤如下：

(1) 输入样本特征，计算协方差矩阵 S，初始化 $\mathrm{W}=\mathrm{S}+\rho\mathrm{I}$；

(2) 重复 $i=1,2,...,p,1,2,...,p,...$，直到收敛：

①将 W 分块， W_{11} 由除去 W 的第i行和第i列的矩阵组成；将 S 分块， s_{12} 由 S 的第i列除去第i项的向量组成；

②输入 W_{11} 和 s_{12}，通过快速坐标下降法求解Lasso问题(2-8)，得到一个 $p-1$ 维向量解 $\hat{\beta}$，将 $\hat{\beta}$ 储存在矩阵 $\hat{\mathrm{B}}$ 的第i列中；

(3)更新 $\boldsymbol{\omega}_{12}=\mathrm{W}_{11}\hat{\boldsymbol{\beta}}$，填入 W 相应的行和列；

(3) 输出最终的协方差矩阵；

(4) 利用收敛后的 $\hat{\mathrm{B}}$ 和 W 求逆协方差矩阵 $\hat{\boldsymbol{\Theta}}$，对于 $i=1,2,...,p$，$\hat{\beta}$ 分别取矩阵 $\hat{\mathrm{B}}$

4. MATLAB程序

中第i列元素，分别计算$\hat{\theta}_{12}=-\hat{\beta}\theta_{22}$和$\hat{\theta}_{22}=1/(\omega_{22}-\omega_{12}^{T}\hat{\beta})$；

(5) 输出估计的逆协方差矩阵。

数据见参考文献[2]

4.1 方法一

demo.m

```
load SP500
data = normlization(data);
S = cov(data);  %样本协方差
[X, W] = glasso_1(double(S), 0.5);
%X:sigma^(-1), W:sigma
[~, idx] = sort(info(:,3));
colormap gray
imagesc(X(idx, idx) == 0)
axis off

%% Data Normalization
function data = normlization(data)
data = bsxfun(@minus, data, mean(data));
data = bsxfun(@rdivide, data, std(data));
end
```

glasso_1.m

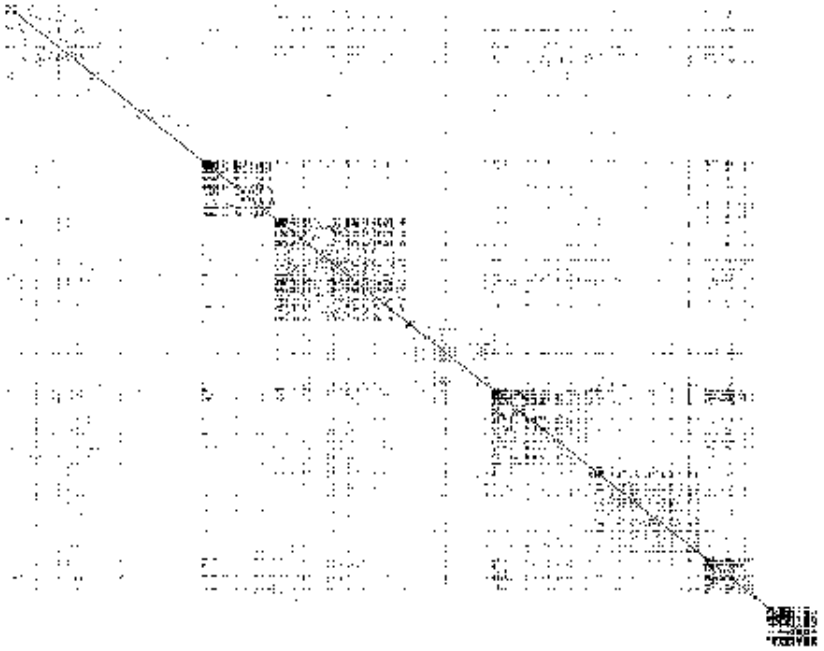
```
function [X, W] = glasso_1(S, lambda)
%% Graphical Lasso - Friedman et. al, Biostatistics, 2008
% Input:
%   S - 样本的协方差矩阵
%   lambda - 罚参数
% Output:
%   X - 精度矩阵    sigma^(-1)
%   W - 协方差矩阵   sigma
%%
p = size(S,1);  %数据维度
W = S + lambda * eye(p);  %W=S+λI
beta = zeros(p) - lambda * eye(p);   %β=-λI
eps = 1e-4;
finished = false(p);   %finished:p*p的逻辑0矩阵
while true
    for j = 1 : p
        idx = 1 : p; idx(j) = [];
        beta(idx, j) = lasso(W(idx, idx), S(idx, j), lambda, beta(idx, j));
        W(idx, j) = W(idx,idx) * beta(idx, j);  %W=W*β
```

```
        W(j, idx) = W(idx, j);
    end
    index = (beta == 0);
    finished(index) = (abs(W(index) - S(index)) <= lambda);
    finished(~index) = (abs(W(~index) -S(~index) + lambda * sign(beta(~index))) < eps);
    if finished
        break;
    end
end
X = zeros(p);
for j = 1 : p
    idx = 1 : p; idx(j) = [];
    X(j,j) = 1 / (W(j,j) - dot(W(idx,j), beta(idx,j)));
    X(idx, j) = -1 * X(j, j) * beta(idx,j);
end
% X = sparse(X);
end
```

lasso.m

```
function w = lasso(A, b, lambda, w)
% Lasso
p = size(A,1);
df = A * w - b;
eps = 1e-4;
finished = false(1, p);
while true
    for j = 1 : p
        wtmp = w(j);
        w(j) = soft(wtmp - df(j) / A(j,j), lambda / A(j,j));
        if w(j) ~= wtmp
            df = df + (w(j) - wtmp) * A(:, j); % update df
        end
    end
    index = (w == 0);
    finished(index) = (abs(df(index)) <= lambda);
    finished(~index) = (abs(df(~index) + lambda * sign(w(~index))) < eps);
    if finished
        break;
    end
end
end
%% Soft thresholding
function x = soft(x, lambda)
x = sign(x) * max(0, abs(x) - lambda);
end
```

结果



注意：罚参数lamda的设定对逆协方差的稀疏性的影响很大，可以用交叉验证方式得到。

4.2 方法二

graphicalLasso.m

```
function [Theta, W] = graphicalLasso(S, rho, maxIt, tol)
% http://www.ece.ubc.ca/~xiaohuic/code/glasso/glasso.htm
% Solve the graphical Lasso
% minimize_{Theta > 0} tr(S*Theta) - logdet(Theta) + rho * ||Theta||_1
% Ref: Friedman et al. (2007) Sparse inverse covariance estimation with the
% graphical lasso. Biostatistics.
% Note: This function needs to call an algorithm that solves the Lasso
% problem. Here, we choose to use to the function *lassoShooting* (shooting
% algorithm) for this purpose. However, any Lasso algorithm in the
% penelized form will work.
%
% Input:
% S -- sample covariance matrix
```

```
% rho --  regularization parameter
% maxIt -- maximum number of iterations
% tol -- convergence tolerance level
%
% Output:
% Theta -- inverse covariance matrix estimate
% W -- regularized covariance matrix estimate, W = Theta^-1

p = size(S,1);

if nargin < 4, tol = 1e-6; end
if nargin < 3, maxIt = 1e2; end

% Initialization
W = S + rho * eye(p);   % diagonal of W remains unchanged
W_old = W;
i = 0;

% Graphical Lasso loop
while i < maxIt,
    i = i+1;
    for j = p:-1:1,
        jminus = setdiff(1:p,j);
        [V D] = eig(W(jminus,jminus));
        d = diag(D);
        X = V * diag(sqrt(d)) * V'; % W_11^(1/2)
        Y = V * diag(1./sqrt(d)) * V' * S(jminus,j);    % W_11^(-1/2) * s_12
        b = lassoShooting(X, Y, rho, maxIt, tol);
        W(jminus,j) = W(jminus,jminus) * b;
        W(j,jminus) = W(jminus,j)';
    end
    % Stop criterion
    if norm(W-W_old,1) < tol,
        break;
    end
    W_old = W;
end
if i == maxIt,
    fprintf('%s\n', 'Maximum number of iteration reached, glasso may not converge.');
end

Theta = W^-1;

% Shooting algorithm for Lasso (unstandardized version)
function b = lassoShooting(X, Y, lambda, maxIt, tol),

if nargin < 4, tol = 1e-6; end
if nargin < 3, maxIt = 1e2; end

% Initialization
[n,p] = size(X);
```

```
if p > n,
    b = zeros(p,1); % From the null model, if p > n
else
    b = X \ Y;  % From the OLS estimate, if p <= n
end
b_old = b;
i = 0;

% Precompute X'X and X'Y
XTX = X'*X;
XTY = X'*Y;

% Shooting loop
while i < maxIt,
    i = i+1;
    for j = 1:p,
        jminus = setdiff(1:p,j);
        S0 = XTX(j,jminus)*b(jminus) - XTY(j);  % S0 = X(:,j)'*(X(:,jminus)*b(jminus)-Y)
        if S0 > lambda,
            b(j) = (lambda-S0) / norm(X(:,j),2)^2;
        elseif S0 < -lambda,
            b(j) = -(lambda+S0) / norm(X(:,j),2)^2;
        else
            b(j) = 0;
        end
    end
    delta = norm(b-b_old,1);    % Norm change during successive iterations
    if delta < tol, break; end
    b_old = b;
end
if i == maxIt,
    fprintf('%s\n', 'Maximum number of iteration reached, shooting may not converge.');
end
```

结果

```
>> A=[5.9436    0.0676    0.5844   -0.0143
    0.0676    0.5347   -0.0797   -0.0115
    0.5844   -0.0797    6.3648   -0.1302
   -0.0143   -0.0115   -0.1302    0.2389
];
>> [Theta, W] = graphicalLasso(A, 1e-4)

Theta =

    0.1701   -0.0238   -0.0159    0.0003
   -0.0238    1.8792    0.0278    0.1034
   -0.0159    0.0278    0.1607    0.0879
    0.0003    0.1034    0.0879    4.2369
```

W =

5.9437	0.0675	0.5843	-0.0142
0.0675	0.5348	-0.0796	-0.0114
0.5843	-0.0796	6.3649	-0.1301
-0.0142	-0.0114	-0.1301	0.2390

5. 补充：近端梯度下降(Proximal Gradient Descent, PGD)求解Lasso问题

L_1 正则化问题的求解可使用近端梯度下降 (Proximal Gradient Descent, 简称 PGD) [Boyd and Vandenberghe, 2004]. 具体来说, 令 ∇ 表示微分算子, 对优化目标

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_1 \ , \tag{11.8}$$

若 $f(\boldsymbol{x})$ 可导, 且 ∇f 满足 L-Lipschitz 条件, 即存在常数 $L > 0$ 使得

$$\left\|\nabla f(\boldsymbol{x}') - \nabla f(\boldsymbol{x})\right\|_2^2 \leqslant L \left\|\boldsymbol{x}' - \boldsymbol{x}\right\|_2^2 \quad (\forall \boldsymbol{x}, \boldsymbol{x}') \ , \tag{11.9}$$

则在 $\boldsymbol{x}_k$ 附近可将 $f(\boldsymbol{x})$ 通过二阶泰勒展式近似为

$$\begin{aligned}\hat{f}(\boldsymbol{x}) &\simeq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|^2 \\ &= \frac{L}{2}\left\|\boldsymbol{x} - \left(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\right)\right\|_2^2 + \text{const}, \end{aligned} \tag{11.10}$$

其中 const 是与 $\boldsymbol{x}$ 无关的常数, $\langle \cdot, \cdot \rangle$ 表示内积. 显然, 式(11.10)的最小值在如下 $\boldsymbol{x}_{k+1}$ 获得:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k) \ . \tag{11.11}$$

于是, 若通过梯度下降法对 $f(\boldsymbol{x})$ 进行最小化, 则每一步梯度下降迭代实际上等价于最小化二次函数 $\hat{f}(\boldsymbol{x})$. 将这个思想推广到式(11.8), 则能类似地得到其每一步迭代应为

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \frac{L}{2}\left\|\boldsymbol{x} - \left(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\right)\right\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \ , \tag{11.12}$$

即在每一步对 $f(\boldsymbol{x})$ 进行梯度下降迭代的同时考虑 L_1 范数最小化.

对于式(11.12), 可先计算 $\boldsymbol{z} = \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)$, 然后求解

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \ . \tag{11.13}$$

令 x^i 表示 $\boldsymbol{x}$ 的第 i 个分量, 将式(11.13)按分量展开可看出, 其中不存在 $x^i x^j$ $(i \neq j)$ 这样的项, 即 $\boldsymbol{x}$ 的各分量互不影响, 于是式(11.13)有闭式解

$$x_{k+1}^i = \begin{cases} z^i - \lambda/L, & \lambda/L < z^i \ ; \\ 0, & |z^i| \leqslant \lambda/L \ ; \\ z^i + \lambda/L, & z^i < -\lambda/L \ , \end{cases} \tag{11.14}$$

其中 x_{k+1}^i 与 z^i 分别是 $\boldsymbol{x}_{k+1}$ 与 $\boldsymbol{z}$ 的第 i 个分量. 因此, 通过 PGD 能使 LASSO 和其他基于 L_1 范数最小化的方法得以快速求解.

6. 参考文献

[1] 林祝莹. 图Lasso及相关方法的研究与应用[D]. 燕山大学, 2016.

[2] Graphical Lasso for sparse inverse covariance selection

[3] 周志华. 机器学习[M]. 清华大学出版社, 2016.

[4] Graphical lasso in R and Matlab

[5] Graphical Lasso