Lecture 15: Advanced topics

Felix Held, Mathematical Sciences

MSA220/MVE440 Statistical Learning for Big Data

28th May 2020

Improving the Lasso

Recall the lasso

The **lasso estimator** of a linear regression problem is the solution to

$$\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \frac{1}{2n} \sum_{l=1}^{n} ||y_l - \mathbf{x}_l^\top \boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1$$

For orthogonal predictors, i.e. $\mathbf{X}^\top \mathbf{X} = \mathbf{I}_p$, we have an **analytical solution**

$$\hat{\boldsymbol{\beta}}^{(j)}_{\text{Lasso}}(\lambda) = \text{sign}\left(\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}}\right)\left(|\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}}| - \lambda\right)_+ = \text{ST}\left(\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}}, \lambda\right)$$

where $(x)_+ = x$ if $x > 0$ and zero otherwise.

The lasso performs **variable selection** by setting some entries $\hat{\boldsymbol{\beta}}^{(i)}_{\text{Lasso}} = 0$, thereby using only those features with non-zero coefficients in $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ for prediction.

The ideal case

Oracle procedure

Assume the true subset of non-zero coefficients is $\mathcal{A} = \{j : \boldsymbol{\beta}_{\text{true}}^{(j)} \neq 0\}$. An oracle procedure leads to an estimator $\hat{\boldsymbol{\beta}}$ such that

1. the right variables are identified, i.e. $\{j : \hat{\boldsymbol{\beta}}^{(j)} \neq 0\} = \mathcal{A}$.
2. the estimation rate is optimal, i.e. $\sqrt{n}(\hat{\boldsymbol{\beta}}^{\mathcal{A}} - \boldsymbol{\beta}_{\text{true}}^{\mathcal{A}}) \xrightarrow{d} N(\mathbf{0}, \boldsymbol{\Sigma})$ for $n \to \infty$ where $\hat{\boldsymbol{\beta}}^{\mathcal{A}}$ is the restriction of $\hat{\boldsymbol{\beta}}$ to elements with indices in $\mathcal{A}$.

Note that these two conditions in particular imply that $\mathbb{E}[\hat{\boldsymbol{\beta}}] \to \boldsymbol{\beta}_{\text{true}}$ for $n \to \infty$.

Does the lasso produce an oracle estimator? Unfortunately (in general) **not**.

- In general $\{j : \hat{\boldsymbol{\beta}}_{\text{Lasso}}^{(j)} \neq 0\} \neq \mathcal{A}$ even for $n \to \infty$ and
- $\hat{\boldsymbol{\beta}}_{\text{Lasso}} \nrightarrow \boldsymbol{\beta}_{\text{true}}$ for $n \to \infty$ even though $\sqrt{n}(\hat{\boldsymbol{\beta}}_{\text{Lasso}} - \boldsymbol{\beta}_{\text{true}})$ converges in distribution in most cases

Finding an oracle estimator

It can be shown that the issues with the lasso arise from the penalisation of the residual sum of squares by $||\boldsymbol{\beta}||_1$.

It can be argued (Fan and Li, 2001) that an **ideal penalty function** should have the following properties

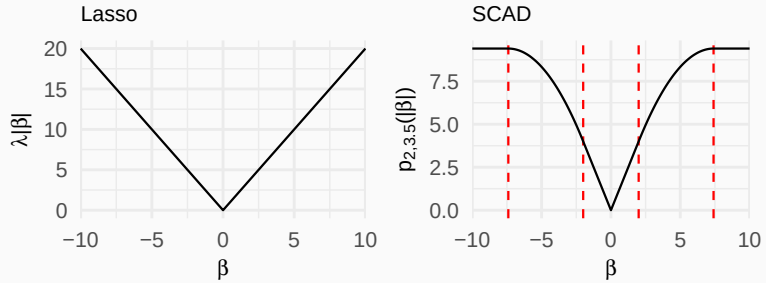
- singularity at zero, leading to sparsity,
- no penalisation of large coefficients, leading to unbiased estimates away from zero,
- differentiability away from zero, and
- convexity.

The **smoothly clipped absolute deviation (SCAD)** penalty combines all these **except convexity**.

Smoothly clipped absolute deviation (SCAD) penalty

The penalty is defined by its derivative $p'_{\lambda,a}(\theta) = \lambda\left(\mathbb{1}(\theta \leq \lambda) + \frac{(a\lambda-\theta)_+}{(a-1)\lambda}\,\mathbb{1}(\theta > \lambda)\right)$ for $\theta > 0$, $\lambda \geq 0$, and $a > 2$. This integrates to

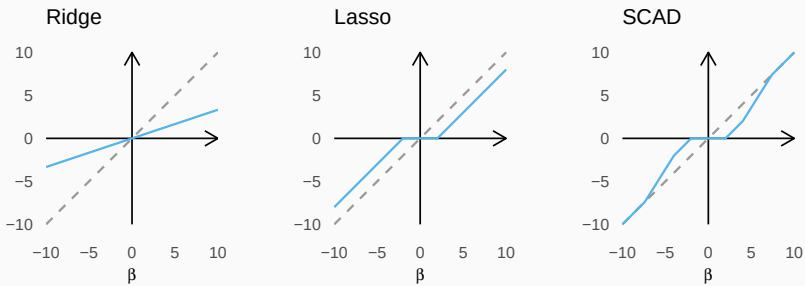
$$p_{\lambda,a}(\theta) = \begin{cases} \lambda\theta & 0 < \theta \leq \lambda \\ -\frac{\theta^2 - 2a\lambda\theta + \lambda^2}{2(a-1)} & \lambda < \theta \leq a\lambda \\ \frac{(a+1)\lambda^2}{2} & \theta > a\lambda \end{cases}$$



SCAD penalty and linear regression

If the SCAD penalty is used to replace the ℓ_1 penalty in the lasso problem and features are orthogonal, then the coefficients can be analytically computed as

$$\hat{\boldsymbol{\beta}}^{(j)}_{\text{SCAD}} = \begin{cases} \text{ST}(\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}}, \lambda) & |\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}}| \leq 2\lambda \\ \left((a-1)\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}} - \text{sign}(\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}})a\lambda\right)/(a-2) & 2\lambda < |\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}}| \leq a\lambda \\ \hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}} & |\hat{\boldsymbol{\beta}}^{(j)}_{\text{OLS}}| > a\lambda \end{cases}$$



Summary: SCAD penalty

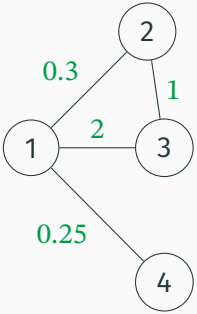
- **Good news**
  - The SCAD penalty gets rid of bias for larger coefficients, but also leads to a sparse solution vector.
  - Under some theoretical conditions on the size of λ as $n \to \infty$ it can be shown that the SCAD penalised linear regression problem is a **oracle procedure**
- **Bad news**
  - The penalty is not convex and standard optimization approaches cannot be used. The authors of the method (Fan and Li, 2001) proposed an algorithm based on local approximations.

Spectral clustering

Starting point

- Many clustering methods focus on **global behaviour** of the data (e.g. GMM, k-means, hierarchical clustering with complete linkage)
- To adapt to **local behaviour** hierarchical clustering with single linkage and the group of **density-based** algorithms (e.g. DBSCAN) showed some success
- In dimension reduction **building a graph of the data** based on k nearest neighbours helped to capture **local behaviour** (e.g. Isomap)
- **Idea:** Build a graph representing local behaviour in the data and find good cut points to partition the graph into K clusters.

Graphs and adjacency matrices



An **adjacency matrix** $\mathbf{A} \in \{0,1\}^{n \times n}$ describes edges between n nodes such that $\mathbf{A}^{(i,j)} = 1$ when there is an edge between nodes i and j and zero otherwise.

In addition, **weights** can be added to the edges, leading to a **weighted adjacency matrix** $\mathbf{W} \in [0,\infty)^{n \times n}$.

For the graph on the left

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{W} = \begin{pmatrix} 0 & 0.3 & 2 & 0.25 \\ 0.3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \end{pmatrix}$$

Note: Undirected graphs have symmetric adjacency matrices. Directed graphs can be described by unsymmetric adjacency matrices.

A graph from data

Recall: In the first step of Isomap, a **weighted undirected graph** was built based on the k nearest neighbours of a data point.

A weighted undirected graph can be constructed from a **weighted adjacency matrix** $\mathbf{W}$.

1. For a data point $\mathbf{x}_l$, find the k nearest neighbours.
2. Set $\mathbf{W}^{(l,l_i)} = g(||\mathbf{x}_l - \mathbf{x}_{l_i}||_2)$ where $g : [0,\infty) \to [0,\infty)$ is a monotone function and $\mathbf{x}_{l_i}$, $i = 1,\dots,k$ are the nearest neighbours of $\mathbf{x}_l$. In addition, set all $\mathbf{W}^{(l,m)} = 0$ for $m \notin \{l_1,\dots,l_k\}$ (in particular $\mathbf{W}^{(l,l)} = 0$).
3. Construct a graph where each node represents a data point $\mathbf{x}_l$ and there is a weighted edge between $\mathbf{x}_l$ and $\mathbf{x}_m$ if $\mathbf{W}^{(l,m)} > 0$.

In Isomap $g(z) = z$, but in the following $g_c(z) = \exp(-z^2/c)$ for $c > 0$.

Node degree and the graph Laplacian

Given the weighted adjacency matrix $\mathbf{W}$, the **degree of node** l describes how well-connected a node is

$$d_l = \sum_{m=1}^{n} \mathbf{W}^{(l,m)}$$

and the **degree matrix** is $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_n)$.

Define now the **graph Laplacian**, a measure of **information flow**, as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

Interpretation: If heat were to be distributed from node to node with flow speeds described by $\mathbf{W}$, then $\mathbf{L}$ takes the role of the discretised **Laplacian operator** ∇^2 in the **heat equation** for the heat distribution $\boldsymbol{\phi}$

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} + k\mathbf{L}\boldsymbol{\phi} = \mathbf{0}$$

Graph cutting

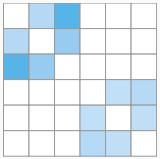
A **good separation of the graph** into two parts A and B is one where flow between the parts is minimized and neither is chosen too small, i.e.

$$\min_{A,B}\left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)\sum_{l\in A,m\in B}\mathbf{W}^{(l,m)}$$

where $\quad \mathrm{vol}(A)=\sum_{l\in A}\sum_{m=1}^{n}\mathbf{W}^{(l,m)}=\sum_{l\in A}d_l$

Raw data and graph

Graph edge weights



Clustering result

Finding good cut points

Finding the best cut point would require to check all possible cuts and is an **NP-hard combinatorial problem**.

Observations and theorem

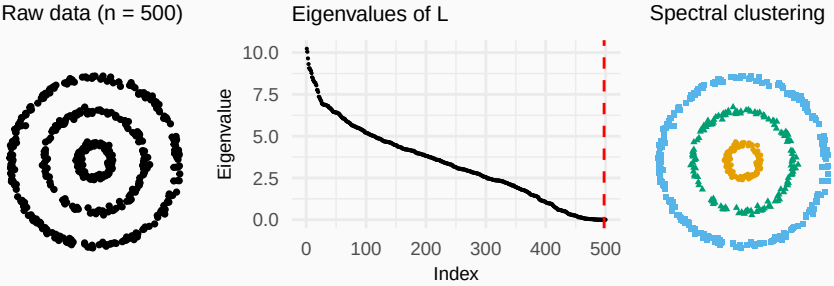
1. The graph Laplacian is symmetric and positive semi-definite, since $\mathbf{y}^\top \mathbf{L}\mathbf{y} = \sum_{i,j=1}^{n} \mathbf{W}^{(i,j)}(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})^2 \geq 0$ for all $\mathbf{y} \in \mathbb{R}^n$.
2. If there are K connected components of the graph, then the set of eigenvectors of $\mathbf{L}$ with eigenvalue 0 is spanned by $\mathbf{1}_{A_k}$ for $k = 1, \ldots, K$, where $\mathbf{1}_{A_k}^{(i)} = 1$ if $i \in A_k$ and zero otherwise.

In practice

- There will not be K separate connected components
- However, if K clusters exist, the smallest K eigenvalues will be near zero and the and corresponding eigenvectors close to indicator vectors.

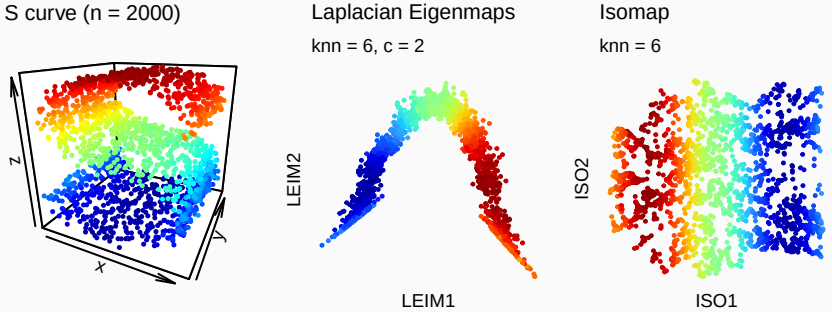
Spectral Clustering

1. Determine the weighted adjacency matrix $\mathbf{W}$ and the graph Laplacian $\mathbf{L}$
2. Find the K smallest eigenvalues of $\mathbf{L}$ that are near zero and well separated from the others
3. Find the corresponding eigenvectors $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_K) \in \mathbb{R}^{n \times K}$ and use k-means on the rows of $\mathbf{U}$ to determine cluster membership



Laplacian Eigenmaps for dimension reduction

- In addition to clustering, the eigenvectors of the Laplacian can also be used for **dimension reduction**.
- For each component, use the q eigenvectors corresponding to the q smallest non-zero eigenvalues as an embedding of the original data.
- Laplacian Eigenmaps can be shown to **optimally preserve the local behaviour on average**, but not necessarily global behaviour.



Network graphs

Network Graphs

Graphs can be used to describe **networks** between different abstract objects. Given some data for a set of variables (the **nodes**) it is often of interest to estimate the best corresponding graph.

Some typical examples are

- Links on websites
- Co-authorship of scientific articles
- Protein interaction networks
- Friends/followers in social networks

Types of graphs for networks

To describe data, the following types of graphs are often used

- **Correlation graph:** Undirected edges weighted by the correlation between variables.
  Caveat: Correlation can be due to a common ancestor.
- **Partial correlation graph:** Undirected edges weighted by the correlation between variables **given all other variables**. This measures how much correlation is left once all other variables are controlled for.
- **Directed acyclic graphs:** Weighted directed edges without cycles describing causality between nodes.

As usual in statistics, recall that

Correlation does not imply causality.

Partial correlation graphs

Assume there are p features in a dataset and represent each feature by a node in a graph. Denote the nodes as $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(p)}$, where $\mathbf{x}$ is a random feature vector before data is observed.

The weight of the edge between variables $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ is their **partial correlation**

$$\mathrm{Corr}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)} | \mathbf{x}^{(k)}, k \neq i, j) = \rho^{(i,j)} = \rho^{(j,i)}$$

- If $\rho^{(i,j)} = 0$ there is no edge between $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$.
- $\rho^{(i,j)}$ captures the information left after controlling for $\mathbf{x}^{(k)}$ for $k \neq i, j$, i.e. the correlation that cannot be explained through a common ancestor

Partial correlation and the normal distribution

Assume now that feature vectors are distributed as

$$\mathbf{x} \sim N(\mathbf{0}, \mathbf{\Sigma})$$

then $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ is called the **precision matrix**. It can be shown that

- $\rho_{ij} = -\dfrac{\mathbf{\Omega}^{(i,j)}}{\sqrt{\mathbf{\Omega}^{(i,i)}\mathbf{\Omega}^{(j,j)}}}$
- With $\mathbf{x}^{(-i)} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(i-1)}, \mathbf{x}^{(i+1)}, \dots, \mathbf{x}^{(p)})$

$$p(\mathbf{x}^{(i)}|\mathbf{x}^{(-i)}) = N\left(-\sum_{j \neq i} \frac{\mathbf{\Omega}^{(i,j)}}{\mathbf{\Omega}^{(i,i)}}\mathbf{x}^{(j)}, \frac{1}{\mathbf{\Omega}^{(i,i)}}\right)$$

It is therefore enough to estimate the precision matrix and show that if $0 = \mathbf{\Omega}^{(i,j)} = \rho_{ij}$ then there is no dependence of $\mathbf{x}^{(i)}$ on $\mathbf{x}^{(j)}$, and vice versa.

Estimating the precision matrix

It can be shown that the log-likelihood of the precision matrix $\mathbf{\Omega}$ given the empirical covariance matrix is

$$l(\mathbf{\Omega}) = \log(|\mathbf{\Omega}|) - \mathrm{tr}(\hat{\mathbf{\Sigma}}\mathbf{\Omega})$$

- Can be used to estimate $\mathbf{\Omega}$ with iterative methods.
- In general, all entries will be non-zero and therefore all edges will be present in the resulting network.

Precision matrix and graph structure

1. For a **known graph structure Γ** where $\Gamma_{ij} \in \{0, 1\}$ the constrained problem

$$\arg\min_{\boldsymbol{\Omega}} -l(\boldsymbol{\Omega}) \quad \text{subject to} \quad \omega_{ij} = 0 \Leftrightarrow \gamma_{ij} = 0$$

   can be solved.

2. If the **graph structure** is **unknown** lasso regularisation can help to uncover relevant edges. This leads to

$$\arg\min_{\boldsymbol{\Omega}} -l(\boldsymbol{\Omega}) + \lambda \sum_{i<j} \left|\boldsymbol{\Omega}^{(i,j)}\right|$$

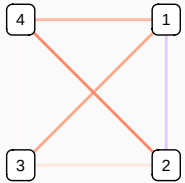
   which can be solved with neighbourhood regression-based lasso or gradient-based lasso. This is called the **Graphical lasso (glasso)**.

Example of network estimation

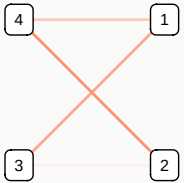
Assume the following empirical covariance matrix and graph structure

$$\widehat{\mathbf{\Sigma}} = \begin{pmatrix} 10 & 1 & 5 & 4 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{pmatrix} \quad \text{and} \quad \mathbf{\Gamma} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

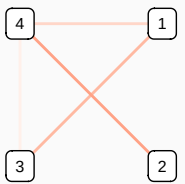
Direct estimate



Known graph structure



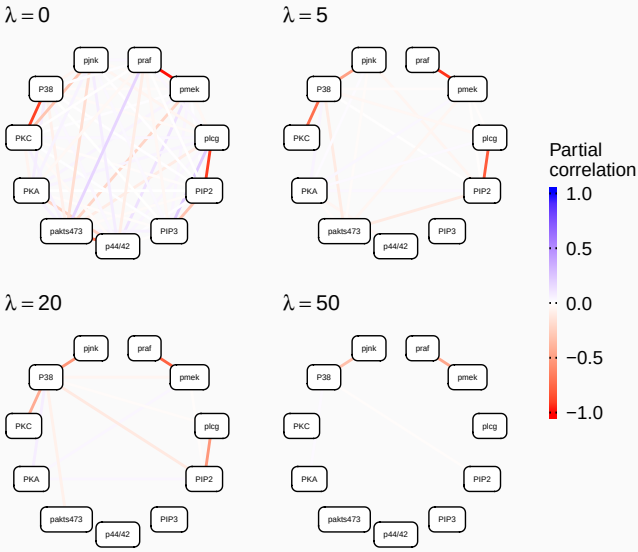
Glasso estimate ($\lambda = 1$)



Graphical lasso: More advanced example

Protein flow cytometry data
($n = 7466$ cells,
$p = 11$ proteins)

Single network estimates from runs of the glasso for increasing λ



Notes on Graphical lasso

- **Warning:** Network estimation with the glasso is unfortunately notoriously unstable, requiring e.g. repeated estimation on bootstrapped samples of the data
- As with any lasso method, the two main caveats are
  1. If too many variables are highly correlated, the network graph cannot be identified
  2. Is the true data generating process sparse?
- Networks are often very interesting to analyse and can reveal a lot about the relationship of variables

Take-home message

- Graphs are very useful tools that can be used for (among other things) dimension reduction, clustering and correlation estimation
- The lasso has short-comings that can be addressed by modifying the ℓ_1 penalty
- Lasso-like techniques can help to estimate sparse networks helping in the interpretation of complex datasets