10-708 PGM (Spring 2020): Homework 3

Andrew ID: changshi
Name: Chang Shi
Collaborators: [Andrew IDs of all collaborators, if any]

1 LSTM-CRF (Xun) (40 points)

In Homework 1, we have implemented the EM updates for HMM, including the forward-backward message passing algorithm for exact posterior inference of the latent states. But everything were discrete, linear, and Gaussian, i.e., boring. We will see a much more interesting version of it, where the features (potential functions) are learned from deep learning.

This time, we are concerned with the supervised sequence tagging problem. In particular, we are provided with the sequence of observations $\mathbf{x}_{1:T}$ and its corresponding labels (tags) $\mathbf{z}_{1:T}$ for training. Our goal is to be able to predict the tags $\mathbf{z}_{1:T}$ for unseen sequence of observations $\mathbf{x}_{1:T}$ at test time.

For instance, consider the named entity recognition (NER) problem. We would like to locate and label named entities mentioned in the unstructured text as predefined categories such as "name of the person" or "name of the place". An example sentence and tagging would be:

Jim	bought	300	shares	of	Acme	Corp.	in	2006.
B-Person	O	O	O	O	B-Org	I-Org	O	B-Time

where three categories considered are $\{\texttt{Person}, \texttt{Org}, \texttt{Time}\}$, and the $\{\texttt{B}, \texttt{I}, \texttt{O}\}$ follows the IOB2 format. For further information, please refer to the Wikipedia page on insideâ"outsideâ"beginning. But you can solve this problem without knowing what IOB2 is.

A workhorse in sequence tagging is the conditional random field (CRF) model. For this problem, consider a linear chain CRF with T time steps, M discrete states, and K-dimensional observations, where $\mathbf{z}_t \in \{0,1\}^M$, $\|\mathbf{z}_t\| = 1$, $\mathbf{x}_t \in \mathbb{R}^K$ for $t \in [T]$.

The reduced graph for the conditional is again a simple chain:

$$\mathbf{z}_1 \text{ —— } \mathbf{z}_2 \text{ —— } \cdots \text{ —— } \mathbf{z}_T$$

with corresponding Gibbs distribution given by

$$p(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \frac{1}{Z(\mathbf{x}_{1:T})} \cdot \tilde{p}(\mathbf{z}_{1:T}), \tag{1}$$

$$\tilde{p}(\mathbf{z}_{1:T}) = \psi_1(\mathbf{z}_1) \cdot \prod_{t=2}^{T} \psi_t(\mathbf{z}_{t-1}, \mathbf{z}_t) \cdot \psi_{T+1}(\mathbf{z}_T), \tag{2}$$

$$Z(\mathbf{x}_{1:T}) = \sum_{\mathbf{z}_{1:T}} \tilde{p}(\mathbf{z}_{1:T}) \tag{3}$$

where the clique potentials are

$$\psi_1(\mathbf{z}_1) = \phi_1(\mathbf{z}_1, \mathbf{x}_{1:T}) \tag{4}$$

$$\psi_t(\mathbf{z}_{t-1}, \mathbf{z}_t) = \phi_t(\mathbf{z}_t, \mathbf{x}_{1:T})\eta_t(\mathbf{z}_{t-1}, \mathbf{z}_t) \quad t = 2, \ldots, T \tag{5}$$

$$\psi_{T+1}(\mathbf{z}_T) = 1 \tag{6}$$

In the CRF nomenclature, ϕ_t is the feature function and η_t is the transition score. In the good old days, humans used to design various rules to generate feature functions. For instance, one intuition is that whether the first character of a word is capitalized is correlated with whether the word is a name of a person. Then a hand-crafted feature function would simply be $\phi(z_t, \mathbf{x}_{1:T}) = w \cdot \mathbf{1}\{z_t = \texttt{Person}, \texttt{first character of } x_t \texttt{ capitalized}\}$, with the linear coefficient w measuring the strength of this feature. Learning the CRF is essentially estimating these parameters given training data.

However, it's 2020 and it is not hard to imagine an obvious way of automatically generating features: neural networks. In particular, for this problem, we will use a bi-directional word-level LSTM, whose input is the sequence $\mathbf{x}_{1:T}$ and output is the singleton potentials $\phi_{1:T}$. The entire LSTM and CRF model will then be trained jointly using stochastic gradient descent with gradients computed by automatic differentiation.

Let θ be the set of parameters of $\{\phi_{1:T}\}$ and $\{\eta_{1:T}\}$. The MLE is then given by

$$\min_{\theta} \ \frac{1}{n} \sum_{i=1}^{n} -\log p_\theta(\mathbf{z}_{1:T}^{(i)} | \mathbf{x}_{1:T}^{(i)}) \tag{7}$$

Similar to HMM, this requires computing the log-partition function, i.e., inference on $\mathbf{z}_{1:T}$. Also similar to HMM, the reduced graph is a tree, hence belief propagation can perform efficient exact inference.

At test time, we would like to find the best tags for a given observation sequence:

$$\max_{\mathbf{z}_{1:T}} \ \log p_\theta(\mathbf{z}_{1:T} | \mathbf{x}_{1:T}) \tag{8}$$

Again, this max-decoding problem can be solved efficiently using the message passing algorithm.

In this problem, we will implement both log-partition function and max-decoding. You should convince yourself that the algorithm can be derived as below. (No need to show.) Please complete the provided code template and submit to Gradescope. The template has a toy problem to play with. The submitted code will be tested against randomly generated problem instances.

Algorithm 1 Negative log-likelihood for CRF

1. Forward messages in log-scale:

$$\log \alpha(\mathbf{z}_1) = \log \phi_1(\mathbf{z}_1, \mathbf{x}_{1:T}) \tag{9}$$

$$\log \alpha(\mathbf{z}_t) = \log \phi_t(\mathbf{z}_t, \mathbf{x}_{1:T}) + \log \sum_{\mathbf{z}_{t-1}} \exp\{\log \eta_t(\mathbf{z}_{t-1}, \mathbf{z}_t) + \log \alpha(\mathbf{z}_{t-1})\} \quad t = 2, \ldots, T \tag{10}$$

2. Log-partition function:

$$\log Z(\mathbf{x}_{1:T}) = \log \sum_{\mathbf{z}_T} \exp\{\log \alpha(\mathbf{z}_T)\} \tag{11}$$

3. Unnormalized density in log-scale:

$$\log \tilde{p}_\theta(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) = \log \phi_1(\mathbf{z}_1, \mathbf{x}_{1:T}) + \sum_{t=2}^{T} (\log \phi_t(\mathbf{z}_t, \mathbf{x}_{1:T}) + \log \eta_t(\mathbf{z}_{t-1}, \mathbf{z}_t)) \tag{12}$$

4. Negative log-likelihood:

$$-\log p_\theta(\mathbf{z}_{1:T} | \mathbf{x}_{1:T}) = \log Z(\mathbf{x}_{1:T}) - \log \tilde{p}_\theta(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) \tag{13}$$

Algorithm 2 Viterbi decoding for CRF

1. Forward max-product messages in log-scale (and optimal indices):

$$\log \alpha^*(\mathbf{z}_1) = \log \phi_1(\mathbf{z}_1, \mathbf{x}_{1:T}) \tag{14}$$

$$\log \alpha^*(\mathbf{z}_t) = \log \phi_t(\mathbf{z}_t, \mathbf{x}_{1:T}) + \max_{\mathbf{z}_{t-1}} \{\log \eta_t(\mathbf{z}_{t-1}, \mathbf{z}_t) + \log \alpha^*(\mathbf{z}_{t-1})\}, \quad t = 2, \ldots, T \tag{15}$$

$$\alpha^+(\mathbf{z}_t) = \operatorname*{argmax}_{\mathbf{z}_{t-1}} \{\log \eta_t(\mathbf{z}_{t-1}, \mathbf{z}_t) + \log \alpha^*(\mathbf{z}_{t-1})\}, \quad t = 2, \ldots, T \tag{16}$$

2. Max score:

$$\max_{\mathbf{z}_{1:T}} \{\log \tilde{p}(\mathbf{z}_{1:T}, \mathbf{x}_{1:T})\} = \max_{\mathbf{z}_T} \{\log \alpha^*(\mathbf{z}_T)\} \tag{17}$$

$$\mathbf{z}_T^* = \operatorname*{argmax}_{\mathbf{z}_T} \{\log \alpha^*(\mathbf{z}_T)\} \tag{18}$$

3. Backward decoding:

$$\mathbf{z}_{t-1}^* = \alpha^+(\mathbf{z}_t^*), \quad t = T, \ldots, 2 \tag{19}$$

2 Consistency of Lasso (Haohan) (20 points)

Before going into graphical lasso, let's first consider a linear regression problem, with covariates $X \in \mathbb{R}^{n \times p}$ and response $y \in \mathbb{R}^n$. In the high-dimensional setting $n \ll p$, the ordinary least squares (OLS) regression will not generalize, so we need a regularized least squares as our model. We consider one of the most prominent regularized regression models, namely the *Lasso*, as our main tool in this homework problem. Lasso estimates the regression coefficients as

$$\hat{\beta}_{\text{lasso}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \tag{20}$$

where λ is a hyperparameter that governs the strength of the regularization and controls the sparsity of the coefficients identified.

The attachment contains the training data $(X_{\text{train}}, y_{\text{train}})$ and the test data $(X_{\text{test}}, y_{\text{test}})$. You can use your favorite Lasso implementation, such as `sklearn.linear_model.Lasso` or `glmnet` in R. We will use mean squared error (MSE) as the main evaluation metric.

2.1 Warm-up (5 points)

Let's do some warm-ups.

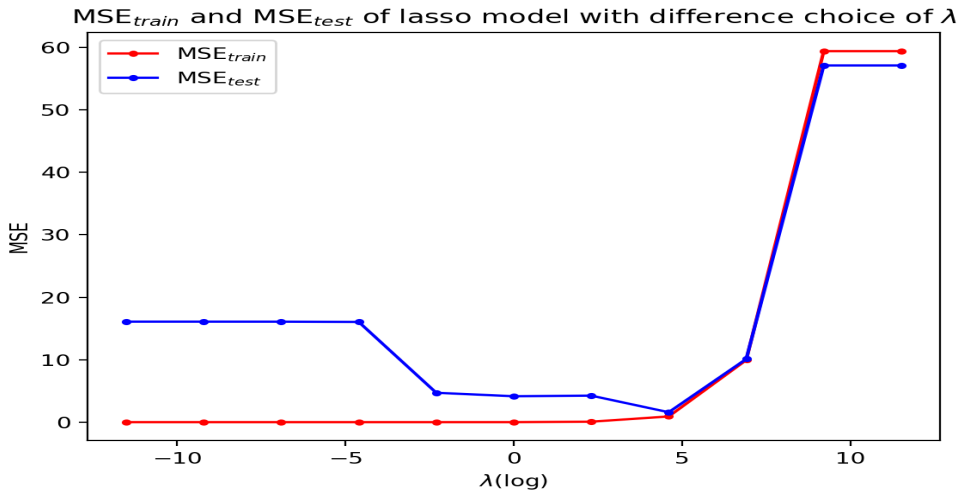
1. First trial (2 points). Fit a Lasso model with $(X_{\text{train}}, y_{\text{train}})$ and test it with $(X_{\text{test}}, y_{\text{test}})$, report $\text{MSE}_{\text{train}}$ and MSE_{test}. You should observe a generalization gap.

   Solution

   With $\lambda = 0.1$, the Lasso model gets $\text{MSE}_{train} = 9.950$ and $\text{MSE}_{test} = 10.107$.

2. Hyperparameter tuning (3 points). Tuning hyperparameters to improve the performance has seemingly become a controversial strategy nowadays. Nonetheless, let's experiment with some choices of λ and check the performance. Please repeat the basic experiment above with 10 choices of λs evenly spaced on a log scale from 10^{-5} to 10^5. Report one plot showing both the $\text{MSE}_{\text{train}}$ and MSE_{test} as a function of λ.

   Solution



2.2 Weak Irrepresentatble Condition (5 points)

You should notice that there is always a generalization gap between training and testing, and the gap seems larger than what can be expected from the measurement errors. Is it some property of the data that has trapped us from closing the generalization gap? The answer is yes.

The data is indeed generated from a linear Gaussian model as follows:

$$y^{(i)} = X^{(i)}\beta^* + \epsilon^{(i)}, \quad \epsilon^{(i)} \sim N(0,1), \quad i = 1, \ldots, n \tag{21}$$

However, with some caveats:

- Only q covariates ($q < p$) are *active*, i.e., associated with the response. In other words, the true $\beta^* \in \mathbb{R}^p$ has q nonzeros.
- For each active covariate j, $\beta_j^* \sim U(0,5)$ and $X_j^{(i)} \sim N(0,1)$ for $i = 1, \ldots, n$.
- What about the rest of the $p - q$ features? In X_{train}, they are duplicates of the active covariates. However, X_{test} is not constructed as so.

Let X_a be the active covariates of X_{train} and X_b be the remaining. We now offer a theoretical tool: if Lasso can correctly identify the active covariates, then

$$|C_{ba}C_{aa}^{-1}\mathbf{1}| < \mathbf{1} \tag{22}$$

where $C_{ba} = \frac{1}{n}X_b^T X_a$, $C_{aa} = \frac{1}{n}X_a^T X_a$, $\mathbf{1}$ denotes a vector of ones, and the inequality holds element-wise.

Show that Lasso cannot correctly identify the active covariates with the data generated as above. It is not required, but please refer to (**?**) for further information.

Solution

let X_i be one nonactive covariate column that appeared in X_b, then $X_a = [X_i, X_{-i}]$, and here we first only consider on column of X_b, so $X_b = [X_i]$.

$$\begin{aligned}
C_{ba} &= \frac{1}{n}X_b^T X_a \\
&= \frac{1}{n}[X_i]^T[X_i, X_{-i}] \\
&= \frac{1}{n}[X_i^T X_i, X_i^T X_{-i}] \\
C_{aa} &= \frac{1}{n}X_a^T X_a \\
&= \frac{1}{n}[X_i, X_{-i}]^T[X_i, X_{-i}] \\
&= \frac{1}{n}\begin{bmatrix} X_i^T X_i & X_i^T X_{-i} \\ X_{-i}^T X_i & X_{-i}^T X_{-i} \end{bmatrix} \\
C_{aa}^{-1} &= n\begin{bmatrix} X_i^T X_i & X_i^T X_{-i} \\ X_{-i}^T X_i & X_{-i}^T X_{-i} \end{bmatrix}^{-1} \\
&= n\begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix}
\end{aligned}$$

Let $\mathbf{A} = X_i^T X_i$,$\mathbf{B} = X_i^T X_{-i}$,$\mathbf{C} = X_{-i}^T X_i$,$\mathbf{D} = X_{-i}^T X_{-i}$. According to the principal of matrix inverse

for block matrix,

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\\ -(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\end{bmatrix}$$

And since $(X_i^T X_i)^{-1}$ is a scalar, we notate it as K, then

$$\begin{aligned}
\mathbf{E} &= K + K^2\mathbf{B}(\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}\mathbf{C}\\
\mathbf{F} &= -K\mathbf{B}(\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}\\
\mathbf{G} &= -K(\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}\mathbf{C}\\
\mathbf{H} &= (\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}\\
|C_{ba}C_{aa}^{-1}\mathbf{1}| &= |[X_i^TX_i, X_i^TX_{-i}]\begin{bmatrix}\mathbf{E} & \mathbf{F}\\ \mathbf{G} & \mathbf{H}\end{bmatrix}\mathbf{1}|\\
&= |[\frac{1}{K}, \mathbf{B}]\begin{bmatrix}\mathbf{E} & \mathbf{F}\\ \mathbf{G} & \mathbf{H}\end{bmatrix}\mathbf{1}|\\
&= |[\frac{1}{K}\mathbf{E}+\mathbf{B}\mathbf{G}, \frac{1}{K}\mathbf{F}+\mathbf{B}\mathbf{H}]\mathbf{1}|\\
\frac{1}{K}\mathbf{E}+\mathbf{B}\mathbf{G} &= \frac{1}{K}(K+K^2\mathbf{B}(\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}\mathbf{C}) + \mathbf{B}(-K(\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}\mathbf{C}) = 1\\
\frac{1}{K}\mathbf{F}+\mathbf{B}\mathbf{H} &= \frac{1}{K}(-K\mathbf{B}(\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}) + \mathbf{B}((\mathbf{D}-K\mathbf{C}\mathbf{B})^{-1}) = \mathbf{0}^T
\end{aligned}$$

Thus,

$$|C_{ba}C_{aa}^{-1}\mathbf{1}| = |[\frac{1}{K}\mathbf{E}+\mathbf{B}\mathbf{G}, \frac{1}{K}\mathbf{F}+\mathbf{B}\mathbf{H}]\mathbf{1}| = |[1, \mathbf{0}^T]\mathbf{1}| = 1$$

The proof above is related to one column of X_b thus one row in the real $|C_{ba}C_{aa}^{-1}\mathbf{1}|$. We then generalize to all column of X_b, so all rows of $|C_{ba}C_{aa}^{-1}\mathbf{1}|$ are equal to one. The (22) inequation does not hold. So Lasso cannot correctly identify the active covariates with the dtata generated as above.

2.3 Improving the Performance (10 points)

It looks like a vanilla Lasso will never solve our problem. Fortunately, we have more knowledge of data. In this section, we will design better methods that take advantage of the knowledge of the data and hopefully get better MSE.

For all the following two questions, please emphasize the design rationale of the method. Regarding empirical performance, please report it in a single plot showing both $\text{MSE}_{\text{train}}$ and MSE_{test} as a function of the hyperparameter. You do not have to stick with Lasso, but please limit yourself within, vaguely, the family of regularized least squares. The grading will significantly value the rationale of the methods than the actual empirical performance since random trial-and-error may also lead to good performance due to the simplicity of the data.

1. Heterogeneity of the samples (5 points). There are rarely truly iid data in the real world and the heterogeneity of the samples often create some spurious signals identified by the model. We offer an extra piece of knowledge of the data:

   - For the remaining $p-q$ covariates of X_{train}, when we create them by duplicating the active covariates, we did not duplicate for every sample, but only 90% of the samples.

   Please take advantage of this message, design a method, test it, and report the performance. Please be creative, but if one needs some inspiration, (**?**) may offer some.

Solution

The stability selection procedure here used is basically iterative sample subsets from the data, fit them with a lasso and find the active covariates by finding the ones with non-zero coefficient. The selection algorithm returns indices of active covariates: $\hat{S}^\lambda = \left\{k : \hat{\beta}_k^\lambda \neq 0\right\}$, and in real implementation, I use $\hat{S}^\lambda = \left\{k : \|\hat{\beta}_k^\lambda\| > 1e^{-4}\right\}$.

The entire algorithms can be described as, each λ do

(a) For each i in 1 ...iter, do:
Sample a subset of data with size ratio r
Run selection algorithm based on lasso coefficient

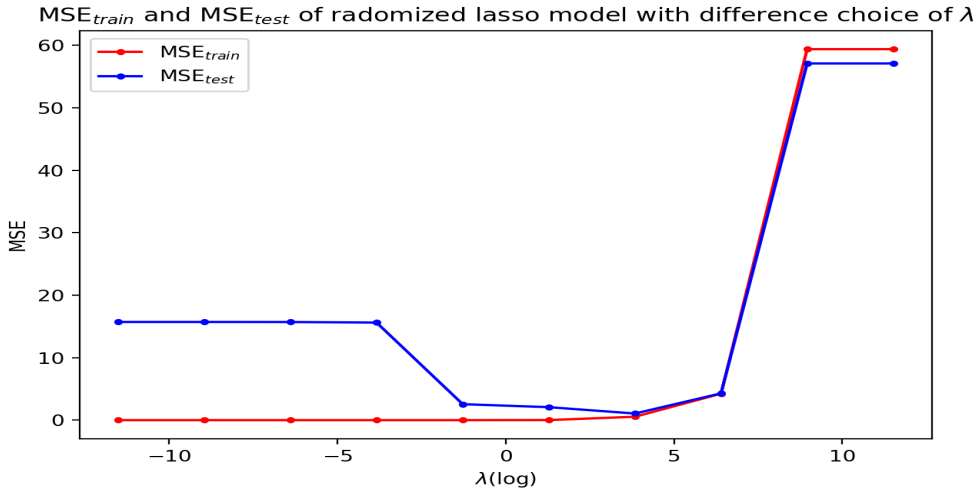
(b) Given the selection set, caclulate the empirical selection probability for each feature:

$$\hat{\Pi}_k^\lambda = \mathbb{P}\left\{k \in \hat{S}^\lambda\right\} = \frac{1}{iter}\sum_{i=1}^{iter} \mathbb{I}\left\{k \in \hat{S}_i^\lambda\right\}$$

If the empirical selection probability is is greater than a threshold π_{thr}, then we assume it is an active covariate.

$$\hat{S}^{\text{stable}} = \left\{k : \hat{\Pi}_k^\lambda \geq \pi_{\text{thr}}\right\}$$

In the implementation, we take $iter = 500$, $r = 0.75$, and $\pi_{thr} = 0.9$. The result is shown as below.



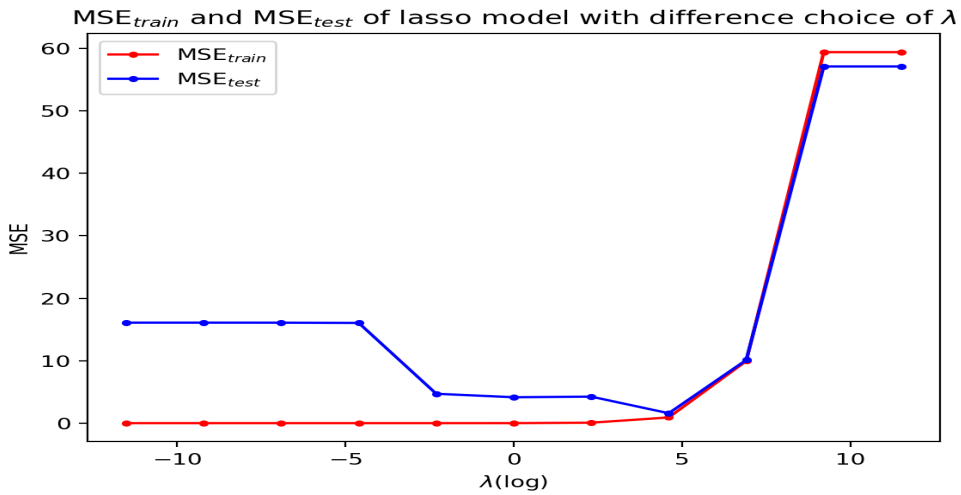
2. Structure of the features (5 points). Another perspective is to take advantage of the knowledge of features, which can often be introduced by some intuitive understanding of the problem in reality. We offer another piece of knowledge of the data:

   - If the i^{th} covariate is active and the j^{th} covariate is its duplicate, then $i < j$.

   Please take advantage of this message, design a method, test it, and report the performance. Please be creative, but if one needs some inspiration, "stepwise selection" may offer some.

Solution

To use the information that "If the i^{th} covariate is active and the j^{th} covariate is its duplicate, then $i < j$", we can loop through the feature columns to find the duplicated covariates, and only select the active covariates, then do lasso. The result is shown as below:



3 Estimation of Graph (Haohan) (40 points)

In Homework 1, we have shown that if a random vector $X \in \mathbb{R}^p$ is jointly Gaussian $X \sim N(0, \Sigma)$, then $(\Sigma^{-1})_{ij} = 0$ if and only if $X_i \perp X_j \mid X_{-ij}$. In this problem, we will use this fact to estimate the conditional independence graph from observations of X using Graphical Lasso (**?**).

The attachment contains the $n \times p$ data matrix. You can use your favorite Graphical Lasso implementation, such as `sklearn.covariance.GraphicalLasso` or `glasso` in R.

1. Logdet program (8 points). Let $S \in \mathbb{R}^{p \times p}$ be the empirical covariance matrix. Show that the MLE is given by the following optimization problem:

$$\min_{\Theta \succeq 0} \ \text{tr}(S\Theta) - \log\det\Theta \tag{23}$$

Solution

$$f(X) = \frac{1}{(2\pi)^{p/2}\det(\Sigma)^{1/2}} \exp\{-\frac{1}{2}X^T\Sigma^{-1}X\} \propto \det(\Theta)^{1/2}\exp\{-\frac{1}{2}X^T\Theta X\}$$

Draw n i.i.d samples $X^{(1)}, \ldots, X^{(n)} \sim N(0, \Sigma)$, then $S = \frac{1}{n}\Sigma_{i=1}^n X^{(i)}X^{(i)T}$ is the empirical covariance matrix, and $\langle S, \Theta\rangle = tr(S\Theta)$. The log likelihood is

$$\ell(\Theta) = \frac{1}{n}\Sigma_{i=1}^n f(X^{(i)}) = \frac{1}{2}\log\det(\Theta) - \frac{1}{2n}\Sigma_{i=1}^n X^{(i)T}\Theta X^{(i)} = \frac{1}{2}\log\det(\Theta) - \frac{1}{2}\langle S, \Theta\rangle$$

Thus Maximum Likelihood Estimation is

$$\max_{\Theta \succeq 0} \frac{1}{2}\log\det(\Theta) - \frac{1}{2}tr(S\Theta)$$

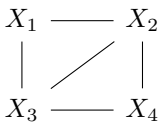
which is the same as

$$\min_{\Theta \succeq 0} \ \mathrm{tr}(S\Theta) - \log\det\Theta$$

2. First run (8 points). Similar to Lasso, in the high-dimensional setting we would like to add an L_1 regularizer, leading to the Graphical Lasso estimate:

$$\min_{\Theta \succeq 0} \ \mathrm{tr}(S\Theta) - \log\det\Theta + \lambda\|\Theta\|_1 \tag{24}$$

Fit Graphical Lasso on the data for $\lambda = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, and report the performance in precision and recall compared to the following ground truth graph:



Solution

I am using the sklearn package of GraphicalLasso implementation here, and I get the results below:

$$\lambda = 10^{-5}, precision = 1.0, recall = 1.0$$
$$\lambda = 10^{-4}, precision = 1.0, recall = 1.0$$
$$\lambda = 10^{-3}, precision = 0.875, recall = 1.0$$
$$\lambda = 10^{-2}, precision = 0.865, recall = 1.0$$
$$\lambda = 10^{-1}, precision = 1.0, recall = 0.857$$

3. Consistency condition (8 points). You probably find that the graphical lasso cannot effectively estimate the graph even if we have 1 million data points for just 4 nodes, and there is no noise in the data at all. Now let's see what's happening.

Again, we offer a theoretical tool to help: it is known that Graphical Lasso can only estimate the graph consistently when the following condition is met:

$$\max_{e \in \mathcal{S}^C} \ \|(\Sigma \otimes \Sigma)_{e,\mathcal{S}}(\Sigma \otimes \Sigma)^{-1}_{\mathcal{S},\mathcal{S}}\|_1 < 1 \tag{25}$$

where $\otimes$ stands for Kronecker product, $\mathcal{S}$ is the support set of the precision matrix (edges in the graph), $\mathcal{S}^C$ is the complement of $\mathcal{S}$, and $(\Sigma \otimes \Sigma)_{e,\mathcal{S}}$ is indexing into a $p^2 \times p^2$ matrix with the first dimension being e and the second dimension being $\mathcal{S}$, resulting in a $1 \times |\mathcal{S}|$ vector.

The data is generated according to the following covariance matrix:

$$\Sigma = \begin{pmatrix} 1 & c & c & 2c^2 \\ c & 1 & 0 & c \\ c & 0 & 1 & c \\ 2c^2 & c & c & 1 \end{pmatrix} \tag{26}$$

with some $c > 0$. Obviously, we must have $c < 1/\sqrt{2}$ to make sure $\Sigma \succeq 0$. Now, since graphical lasso cannot estimate this graph, find a tighter bound of c through numerical experiments.

Solution

$0.208 \leq c < 1/\sqrt{2}$

4. D-trace loss (8 points). Graphical Lasso is not the only way to obtain sparse conditional independence graphs. In fact, the log-determinant term can be computationally demanding. Here's an alternative estimate that does not involve log-det term:

$$\min_{\Theta \succeq 0, \mathrm{diag}(\Theta)=0} \frac{1}{2}\langle \Theta^2, S\rangle - \mathrm{tr}(\Theta) + \lambda \|\Theta\|_1 \tag{27}$$

Show that this new cost function is convex and has a unique minimizer at $\widehat{\Theta} = \Sigma^{-1}$, when $\lambda = 0$.

Solution

When $\lambda = 0$, the D-trace loss function can be donoted as:

$$\ell(\Theta) = \frac{1}{2}\langle \Theta^2, S\rangle - \mathrm{tr}(\Theta)$$

To prove the loss function $\ell(\Theta)$ is convex, we only need to prove that

$$\ell\left(\frac{\Theta_1 + \Theta_2}{2}\right) \leq \frac{\ell(\Theta_1) + \ell(\Theta_2)}{2}$$

$$\ell(\Theta_1) + \ell(\Theta_2) - 2\ell\left(\frac{\Theta_1 + \Theta_2}{2}\right) \geq 0$$

Therefore, we check the following equation :

$$\begin{aligned}
&\ell(\Theta_1) + \ell(\Theta_2) - 2\ell\left(\frac{\Theta_1 + \Theta_2}{2}\right) \\
&= \frac{1}{2}\langle \Theta_1^2, S\rangle - \mathrm{tr}(\Theta_1) + \frac{1}{2}\langle \Theta_2^2, S\rangle - \mathrm{tr}(\Theta_2) \\
&\quad - \langle \left(\frac{\Theta_1 + \Theta_2}{2}\right)^2, S\rangle + 2\mathrm{tr}(\frac{\Theta_1 + \Theta_2}{2}) \\
&= \frac{1}{2}\sum_{i,j} \Theta_{1ij}^2 S_{ij} - \sum_i \Theta_{1ii} + \frac{1}{2}\sum_{i,j} \Theta_{2ij}^2 S_{ij} - \sum_i \Theta_{2ii} \\
&\quad - \sum_{i,j} \left(\frac{1}{2}(\Theta_1 + \Theta_2)\right)_{ij}^2 S_{ij} + \sum_i (\Theta_1 + \Theta_2)_{ii} \\
&= \frac{1}{2}\sum_{i,j} \Theta_{1ij}^2 S_{ij} + \frac{1}{2}\sum_{i,j} \Theta_{2ij}^2 S_{ij} - \sum_{i,j} \left(\frac{1}{2}(\Theta_1 + \Theta_2)\right)_{ij}^2 S_{ij} \\
&= \left\langle \left(\frac{\Theta_1 - \Theta_2}{2}\right)^2, S \right\rangle \\
&\geq 0
\end{aligned}$$

Thus, we proved that the D-trace loss function is convex.
To show that the loss function has a unique minimizer at $\widehat{\Theta} = \Sigma^{-1}$:

$$\frac{d(\ell(\Theta))}{d\Theta} = (\Theta S + S\Theta)/2 - I$$
$$H(L(\Theta)) = (S \otimes I + I \otimes S)/2$$

Since S is positive definite, the Hessian has only positive eigenvalues and so is positive definite. Therefore, the loss function has a unique minimizer at $\widehat{\Theta} = S^{-1}$

5. Consistency condition (8 points). The corresponding condition for this cost function is that:

$$\max_{e \in \mathcal{S}^C} \|\Gamma_{e,\mathcal{S}} \Gamma_{\mathcal{S},\mathcal{S}}^{-1}\|_1 < 1 \tag{28}$$

where $\Gamma = (\Sigma \oplus \Sigma)/2$ and $\oplus$ stands for Kronecker sum. If the provided data satisfies this condition but not the Graphical Lasso condition, find a tighter bound of c through numerical experiments.

Solution

$0.208 \leq c \leq 0.315$

This entire section is constructed based on (**???**).