tfConstrainedGauss Python package

Oliver K. Ernst

September 25, 2021

This package implements two methods for finding a sparse precision matrix with a given structure from a given covariance matrix.

1 Identity-based method

Given an $n \times n$ covariance matrix, here of size $n = 3$:

$$\Sigma = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{pmatrix} \tag{1}$$

and given the structure of the precision matrix (i.e. given the Gaussian graphical model), for example:

$$P = \begin{pmatrix} p_{11} & p_{12} & 0 \\ p_{12} & p_{22} & p_{23} \\ 0 & p_{23} & p_{33} \end{pmatrix} \tag{2}$$

(note that the diagonal elements are always non-zero), the goal is to find the elements of the precision matrix by:

$$P^* = \underset{P}{\operatorname{argmin}} |P\Sigma - I| \tag{3}$$

where I is the identity.

The advantage of this approach is that it does not require calculating the inverse of any matrix, particularly important for large n.

The disadvantage of this approach is that the solution found for P may not yield a covariance matrix P^{-1} whose individual elements are close to those of Σ. That is, while $P\Sigma$ may be close to the identity, there are likely errors in every single element of P^{-1}.

2 MaxEnt-based method

Given the structure of the $n \times n$ precision matrix (i.e. given the Gaussian graphical model), for example:

$$P = \begin{pmatrix} p_{11} & p_{12} & 0 \\ p_{12} & p_{22} & p_{23} \\ 0 & p_{23} & p_{33} \end{pmatrix} \tag{4}$$

(note that the diagonal elements are always non-zero), and given the covariances for corresponding to every *non-zero* entry in P, i.e. given:

$$c_{11}, c_{12}, c_{22}, c_{23}, c_{33} \tag{5}$$

the goal is to find the elements of P. In other words, every unique element (i, j) of the $n \times n$ symmetric matrix has a given constraint, either to a value in the covariance matrix, or a zero entry in the precision matrix.

This is a maximum entropy (MaxEnt) setup. The elements of the precision matrix p_{ij} are directly the interactions in the Gaussian graphical model; the moments they control in a MaxEnt sense are the covariances c_{ij}.

The problem can be solved in a number of ways, for example using Boltzmann machine learning, where we minimize:

$$P^* = \underset{P}{\operatorname{argmin}}\, \mathcal{D}_{\mathcal{KL}} = \min_P \sum_n p(n) \ln \frac{p(n)}{\tilde{p}(n)} \tag{6}$$

where $p(n)$ is the (unknown) data distribution that gave rise to the given covariances c_{ij} and $\tilde{p}(n)$ is the Gaussian with precision matrix P. The gradients that result are the wake sleep phase:

$$\Delta p_{ij} \propto c_{ij} - (P^{-1})_{ij} \tag{7}$$

In TensorFlow, we minimize the MSE loss for the individual terms, which results in the same first order gradients:

$$P^* = \underset{P}{\operatorname{argmin}} \sum_{ij} \left|\left| c_{ij} - (P^{-1})_{ij} \right|\right|_2 \tag{8}$$

To learn each element of the covariance matrix with equal importance, we can use a weighted MSE loss:

$$P^* = \underset{P}{\operatorname{argmin}} \sum_{ij} w_{ij} \left|\left| c_{ij} - (P^{-1})_{ij} \right|\right|_2 \tag{9}$$

where

$$w_{ij} = c_{ij}^{-2} \tag{10}$$