北京工業大學
BEIJING UNIVERSITY OF TECHNOLOGY

基于拓扑交换优化的深度学习脑效应连接学习方法

21071125 易思铭

不息為体 日新為道

北京工業大學
BEIJING UNIVERSITY OF TECHNOLOGY

目 录
CONTENTS

不息为体 日新为道

字太小了，ppt字一般要大一些至少20吧，标题这种应该更大一些

课题介绍

课题介绍

脑效应连接网络是人脑连接研究中一项重要的研究课题，从功能磁共振成像（functional Magnetic Resonance Imaging，fMRI）数据中学习不同脑区间的因果效应连接。而生命科学的研究表明人脑效应连接的变化通常先于人体异常行为症状的显现,所以脑效应连接模式异常的发现，可以给一些脑疾病的早期诊断提供新线索。因此，如何从fMRI时间序列数据中识别脑效应连接网络已成为脑科学项研究中的前沿热点。

随着深度学习的兴起，一些研究者针对fMRI数据的特点，试图采用深度学习的技术手段来获取精度更高的脑效应连接网络。这些深度学习方法一般都将脑效应网络的学习问题转换为有向无环图的非凸优化问题。然而，这些方法常常面临局部最优解以及约束处理复杂的问题。因此，如何设计高效的优化算法，在确保结构无环的同时，提升优化精度和效率，是当前研究中的难点和热点。

统一一下正文字体，比如中文是宋体，英文是 times new roman

而拓扑优化通过节点对的交换操作，在一定范围内搜索更优的有向无环图结构。这种局部搜索方法可以快速调整图的结构，使得优化算法在每一步都能逐步接近更优的局部极小值点，提升整体优化效率。并且拓扑排序方法可以灵活地结合现有的优化技术，作为后处理步骤进一步优化已有的有向无环图学习算法，从而提高其效果。这种灵活性使得拓扑优化成为一种易于集成和应用的工具。因此，本课题拟提出一种基于拓扑交换优化的深度学习脑效应连接学习方法，该方法采用在使用拓扑排序优化方法来改进现有深度学习方法的性能，以期更高效地学习脑效应连接网络。

字大小也统一一下，而且正文行数不要太多，ppt是用来展示核心观点的，不是让你用来写论文的

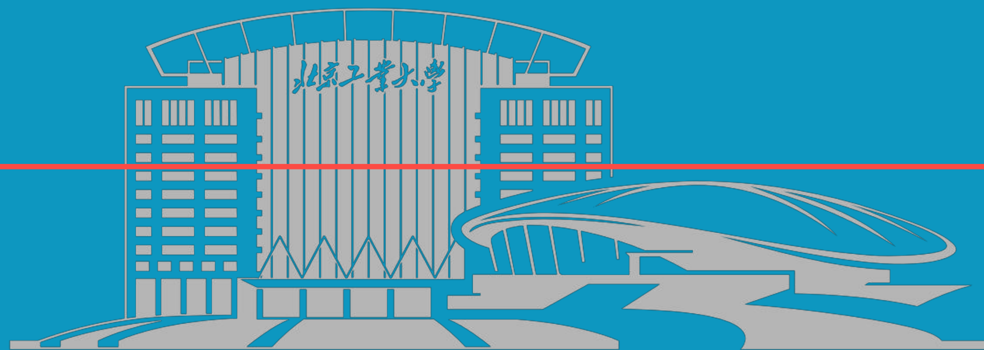
北京工業大學
BEIJING UNIVERSITY OF TECHNOLOGY

不息为体 日新为道

这页的排版不是很好，然后下面的黑字在蓝底上不容易看清

方法论

研究基于两个方法论：
DAGs with NO TEARS: Continuous Optimization for Structure Learning
和
Optimizing NOTEARS Objectives via Topological Swaps



这几页方法论最大的问题就是，排版太乱了，原文的截图感觉不是很必要。

NOTEARS: Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning [1]

用于结构学习的通过轨迹指数和增广拉格朗日的非组合优化方法

3 A new characterization of acyclicity

In order to make (3) amenable to black-box optimization, we propose to replace the combinatorial acyclicity constraint $\mathsf{G}(W) \in \mathbb{D}$ in (3) with a single smooth equality constraint $h(W) = 0$. Ideally, we would like a function $h : \mathbb{R}^{d \times d} \to \mathbb{R}$ that satisfies the following desiderata:

(a) $h(W) = 0$ if and only if W is acyclic (i.e. $\mathsf{G}(W) \in \mathbb{D}$);
(b) The values of h quantify the "DAG-ness" of the graph;
(c) h is smooth;
(d) h and its derivatives are easy to compute.

简单来说，论文作者提出了一个新的函数h(W)（如左图），这个函数仅在图无环时取零。

从而将DAG学习问题变为一个标准的连续优化问题，可以利用诸如L-BFGS等数值算法高效求解。

这里截图中应该是h(w)的特性，或者说需要满足的要求，具体的函数形式是Theorem 1

NOTEARS: Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning [1]

用于结构学习的通过轨迹指数和增广拉格朗日的非组合优化方法

A key conclusion from Theorem 1 is that h and its gradient only involve evaluating the matrix exponential, which is a well-studied function in numerical anlaysis, and whose $O(d^3)$ algorithm [1] is readily available in many scientific computing libraries. Although the connection between trace of matrix power and number of cycles in the graph is well-known [19], to the best of our knowledge, this characterization of acyclicity has not appeared in the DAG learning literature previously. We defer the discussion of other possible characterizations in the appendix. In the next section, we apply Theorem 1 to solve the program (3) to stationarity by treating it as an equality constrained program.

h(W)及其梯度只包含评估矩阵的增长率，这个可以用已经被广泛运用的时间复杂度为O(d³)的算法进行计算。

Algorithm 1 NOTEARS algorithm

1. Input: Initial guess (W_0, α_0), progress rate $c \in (0,1)$, tolerance $\epsilon > 0$, threshold $\omega > 0$.
2. For $t = 0, 1, 2, \ldots$:
   (a) Solve primal $W_{t+1} \leftarrow \arg\min_W L^\rho(W, \alpha_t)$ with ρ such that $h(W_{t+1}) < ch(W_t)$.
   (b) Dual ascent $\alpha_{t+1} \leftarrow \alpha_t + \rho h(W_{t+1})$.
   (c) If $h(W_{t+1}) < \epsilon$, set $\widetilde{W}_{\mathsf{ECP}} = W_{t+1}$ and break.
3. Return the thresholded matrix $\widehat{W} := \widetilde{W}_{\mathsf{ECP}} \circ 1(|\widetilde{W}_{\mathsf{ECP}}| > \omega)$.

NOTEARS算法的具体步骤如左图，使用h(W)函数求解增广拉格朗日函数，再利用对偶更新确保最终获得满足无环性约束的局部最优解。*

*：此部分内容由AI总结

方法论2：Optimizing NOTEARS Objectives via Topological Swaps

和方法论1同样的问题，然后为什么这里的样式变了，其他的地方都是白底黑字就这里是蓝底白字，感觉很割裂

在前一篇论文的基础上，Deng et al.在Optimizing NOTEARS Objectives via Topological Swaps中提出了一种新型的双层算法来优化目标函数。[2]

1. We propose a bi-level optimization algorithm, in which the outer level optimizes over topological orders and the inner level optimizes the score given a specific order. To optimize over orders, we use a novel technique for selecting candidate pairs of nodes to be swapped, which is described in detail in Section 4. This approach involves iteratively swapping pairs of nodes within the topological order of a DAG, and utilizes the KKT conditions as a guide for determining which pairs to consider swapping. To optimize the score given a specific order, we utilize state-of-the-art solvers that are able to solve the problems to stationary points.

The outer level: optimizes over topological orders
外层：通过拓扑交换进行优化

The inner level: optimizes the score given a specific order
内层：通过特定顺序的分数来进行优化。上一篇论文中所提及的 h(W) 被用作一个衡量无环性的光滑约束函数，当 h(W)=0 时，表示当前的加权邻接矩阵对应一个DAG。内层的求解就是在这个条件下找到一个局部最优解。

内层通过h(W)=0判定无环，外层通过拓扑交换提高整体目标函数的优化效率

方法论2：Optimizing NOTEARS Objectives via Topological Swaps

在前一篇论文的基础上，Deng et al.在Optimizing NOTEARS Objectives via Topological Swaps中提出了一种新型的双层算法来优化目标函数。[2]

3. We conduct a comprehensive set of experiments in multiple settings to evaluate the performance of our algorithm against state-of-the-art methods for solving problem (1). The results of our experiments, summarized in Section 5, demonstrate that our method is able to find minimizers with lower scores (compared to existing algorithms) that are guaranteed to be either local minima or KKT points.

作者还应用了KKT条件来判断是否进行节点交换，这样可以确保每次交换目标函数的值，并且也能保证在一定条件之下所获得解是局部最优/满足KKT条件的。

补充：KKT（Karush–Kuhn–Tucker）条件[3][4][5]

Necessary conditions [edit]

Suppose that the objective function $f: \mathbb{R}^n \to \mathbb{R}$ and the constraint functions $g_i: \mathbb{R}^n \to \mathbb{R}$ and $h_j: \mathbb{R}^n \to \mathbb{R}$ have subderivatives at a point $x^* \in \mathbb{R}^n$. If x^* is a local optimum and the optimization problem satisfies some regularity conditions (see below), then there exist constants $\mu_i\ (i = 1, \ldots, m)$ and $\lambda_j\ (j = 1, \ldots, \ell)$, called KKT multipliers, such that the following four groups of conditions hold:[8]

Stationarity

For minimizing $f(x)$:

$$\partial f(x^*) + \sum_{j=1}^{\ell} \lambda_j \partial h_j(x^*) + \sum_{i=1}^{m} \mu_i \partial g_i(x^*) \ni \mathbf{0}$$

For maximizing $f(x)$:

$$-\partial f(x^*) + \sum_{j=1}^{\ell} \lambda_j \partial h_j(x^*) + \sum_{i=1}^{m} \mu_i \partial g_i(x^*) \ni \mathbf{0}$$

Primal feasibility

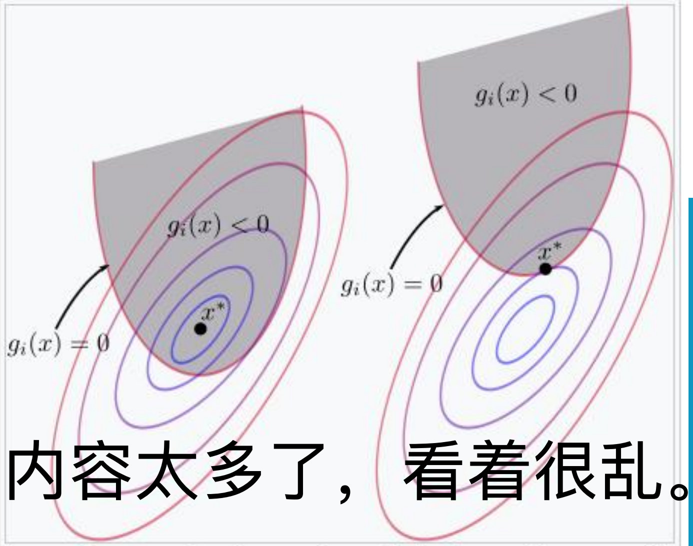
$$h_j(x^*) = 0, \text{ for } j = 1, \ldots, \ell$$
$$g_i(x^*) \le 0, \text{ for } i = 1, \ldots, m$$

Dual feasibility

$$\mu_i \ge 0, \text{ for } i = 1, \ldots, m$$

Complementary slackness

$$\sum_{i=1}^{m} \mu_i g_i(x^*) = 0.$$



Inequality constraint diagram for optimization problems

优化下排版吧，内容太多了，看着很乱。

The last condition is sometimes written in the equivalent form: $\mu_i g_i(x^*) = 0, \text{ for } i = 1, \ldots, m.$

In the particular case $m = 0$, i.e., when there are no inequality constraints, the KKT conditions turn into the Lagrange conditions, and the KKT multipliers are called Lagrange multipliers.

简化版的方程式：

$$\nabla_{\mathbf{x}} L = \nabla f + \lambda \nabla g = \mathbf{0}$$
$$g(\mathbf{x}) \le 0$$
$$\lambda \ge 0$$
$$\lambda g(\mathbf{x}) = 0.$$

从KKT等式约束优化问题而来

约束式
LagRange乘数
互补松弛性

四个KKT条件分别对应：[3]

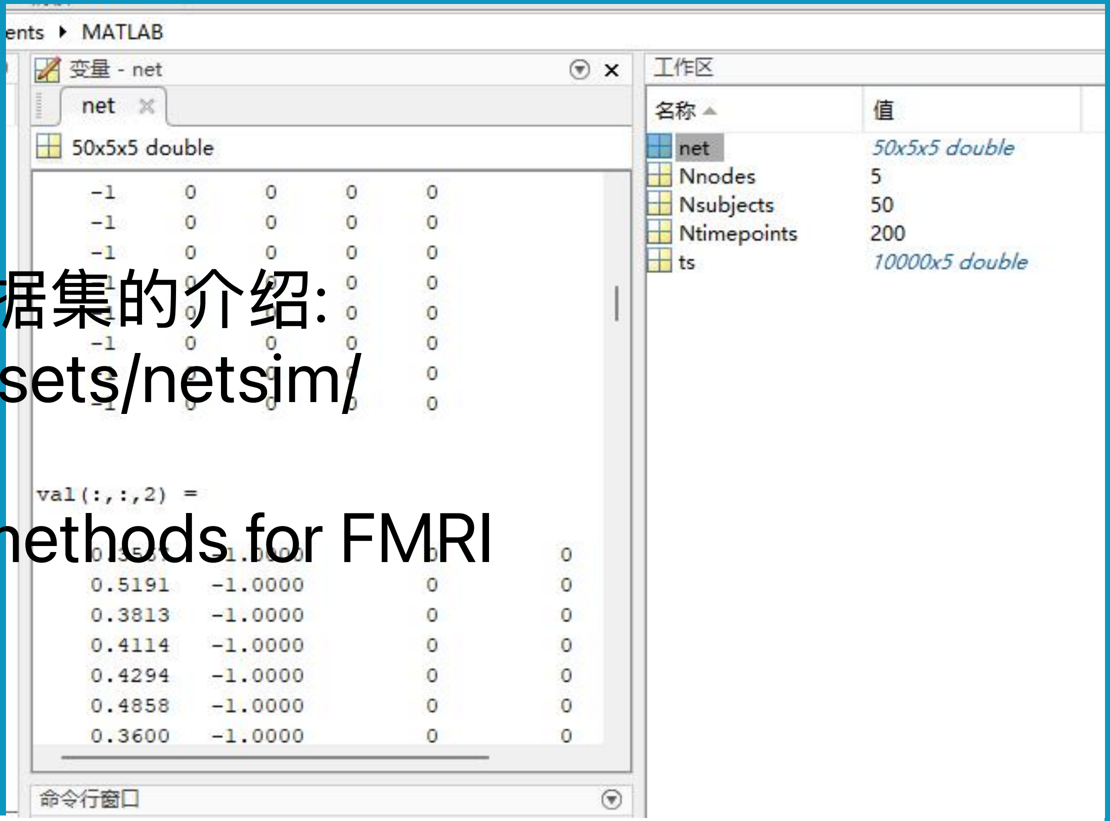
- –最佳解的必要条件包括Lagrangian函数 L(x,λ) 的定常方程式（Stationarity）
- –原始可行性(Primal feasibility)
- –对偶可行性(dual feasibility)
- –互补松弛性(Complementary slackness)

方法论3：两篇论文为什么/怎么应用到脑效应连接中

脑效应连接旨在揭示大脑各区域之间的因果或方向性联系，这本质上可以用有向无环图来建模。

而前面两篇参考文献就是为了解决DAG结构学习问题而设计，把传统离散搜索转化为连续优化问题，并利用拓扑交换在全局上探索更优的拓扑排序，进而提高结构恢复的准确性。

数据集如下图：



这里数据集建议换成对于这个数据集的介绍:
https://www.fmrib.ox.ac.uk/datasets/netsim/
这是那个数据集的网址
相关论文：Network modelling methods for FMRI

当前进度

当前进度：已完成

北京工业大学
BEIJING UNIVERSITY OF TECHNOLOGY

文献阅读&代码实现

文献阅读方面，已经基本完整阅读方法论中所提及的两篇背景论文：

- **DAGs with NO TEARS: Continuous Optimization for Structure Learning**
- **Optimizing NOTEARS Objectives via Topological Swaps**

两篇文章中所实现的代码已经在本地成功部署：（下图为非线性拓扑交换优化**NOTEARS**）

运行　topo_nonlinear　topo_linear

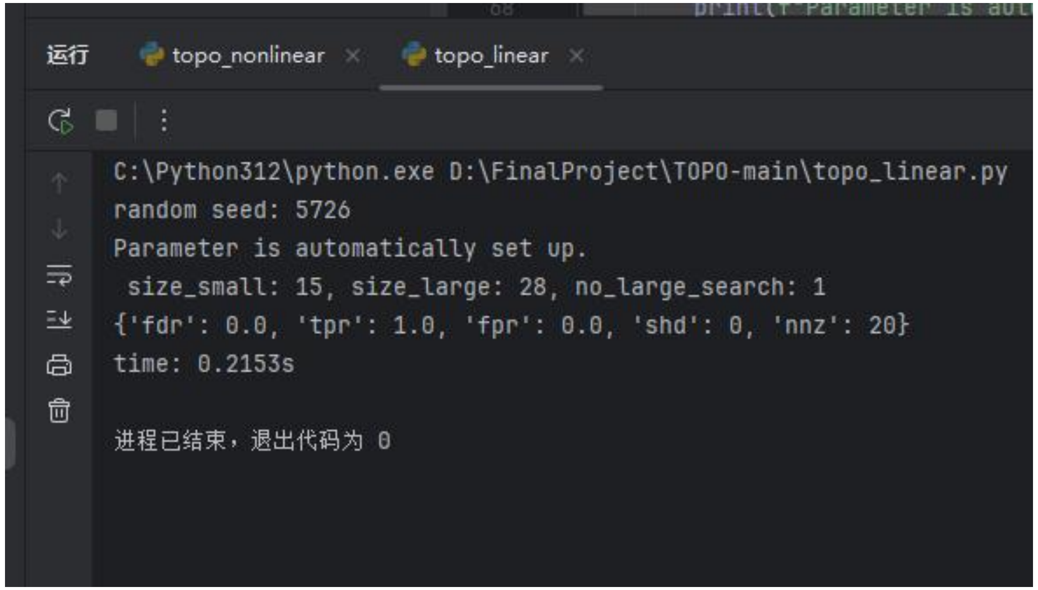
```
working with topological sort:[np.int64(2), np.int64(3), np.int64(4), np.int64(6), np.int64(5), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(0)], current loss 7.222629487152364
working with topological sort:[np.int64(2), np.int64(3), np.int64(4), np.int64(5), np.int64(0), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(6)], current loss 11.856900269099715
working with topological sort:[np.int64(2), np.int64(5), np.int64(4), np.int64(6), np.int64(0), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(3)], current loss 12.280364268671029
working with topological sort:[np.int64(2), np.int64(3), np.int64(9), np.int64(6), np.int64(0), np.int64(4), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 12.667044354703643
working with topological sort:[np.int64(2), np.int64(3), np.int64(0), np.int64(6), np.int64(4), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 7.29417923860849
working with topological sort:[np.int64(2), np.int64(3), np.int64(4), np.int64(6), np.int64(0), np.int64(9), np.int64(7), np.int64(1), np.int64(8), np.int64(5)], current loss 8.143583068651617
working with topological sort:[np.int64(2), np.int64(3), np.int64(6), np.int64(4), np.int64(0), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 8.665887690846715
working with topological sort:[np.int64(3), np.int64(2), np.int64(4), np.int64(6), np.int64(0), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 7.842590146677719
working with topological sort:[np.int64(2), np.int64(9), np.int64(4), np.int64(6), np.int64(0), np.int64(3), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 11.826632527339042
working with topological sort:[np.int64(2), np.int64(3), np.int64(4), np.int64(6), np.int64(0), np.int64(9), np.int64(8), np.int64(7), np.int64(1), np.int64(5)], current loss 7.301786957744236
working with topological sort:[np.int64(2), np.int64(0), np.int64(4), np.int64(6), np.int64(3), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 7.249878432469249
working with topological sort:[np.int64(2), np.int64(3), np.int64(4), np.int64(0), np.int64(6), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 7.539060179261852
working with topological sort:[np.int64(2), np.int64(3), np.int64(4), np.int64(6), np.int64(0), np.int64(9), np.int64(1), np.int64(8), np.int64(7), np.int64(5)], current loss 8.286755833739267
working with topological sort:[np.int64(2), np.int64(6), np.int64(4), np.int64(3), np.int64(0), np.int64(9), np.int64(7), np.int64(8), np.int64(1), np.int64(5)], current loss 7.438340301768617
Using larger search space, but we cannot find better loss
{'fdr': 0.090909, 'tpr': 1.0, 'fpr': 0.028571, 'shd': 1, 'nnz': 11}
running time is 691.1391069889069

进程已结束，退出代码为 0
```

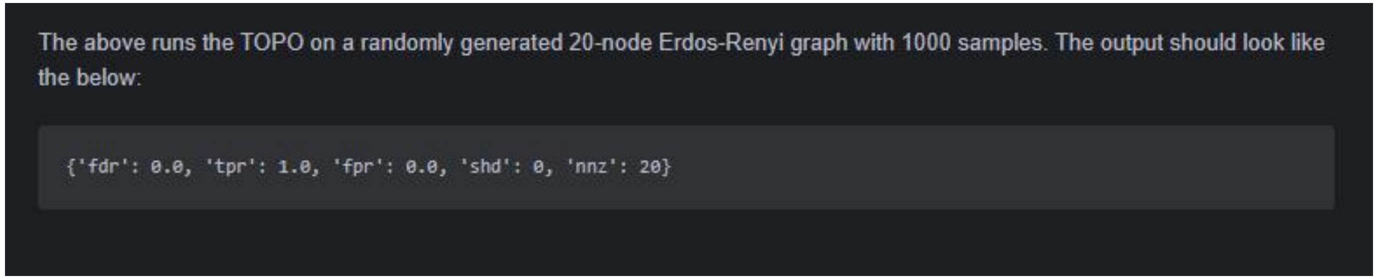
当前进度：已完成

代码实现

下图为线性拓扑优化**NOTEARS**运行结果及论文指导理论实验结果对比

左：本地运行；右：论文指导理论实验结果





当前进度：未完成

你可以总结一下已完成的任务和未完成的任务，这一页可以变成遇到的问题

代码实现&论文撰写

1. 使用脑效应连接的数据集时，结构匹配不合理：

- fdr = 0.5 表示预测的边中有 50% 是错误的
- tpr = 0.2 表示只正确找到了 20% 的真实边
- fpr = 2.0 误报率较高
- shd = 8 与真实图相比存在 8 个结构上的差异
- nnz = 4 最终仅预测了 4 条边

目前猜测可能是数据处理/噪声处理/参数设置上有问题，还在修改中

2.创新点?

目前考虑的是首先完成28个数据集的成功预测，然后可能想从高噪声情况效果不好的方向出发进行优化（如果前面调试顺利的话）

3.论文尚未开始撰写

```
运行    brain_connectivity ×
C:\Python312\python.exe D:\FinalProject\TOPO-main\brain_connectivity.py
单个受试者的 ts shape: (200, 5)
拼接后的 X shape: (10000, 5)
random seed: 2795
Parameter is automatically set up.
 size_small: 6, size_large: 7, no_large_search: 0
{'fdr': 0.5, 'tpr': 0.2, 'fpr': 2.0, 'shd': 8, 'nnz': 4}
time: 0.0721s

进程已结束，退出代码为 0
```

参考文献

[1] Zheng, Xun, et al. "Dags with no tears: Continuous optimization for structure learning." Advances in neural information processing systems 31 (2018).

[2] Deng, Chang, et al. "Optimizing notears objectives via topological swaps." International Conference on Machine Learning. PMLR, 2023.

[3] Karush-Kuhn-Tucker (KKT)条件，知乎

https://zhuanlan.zhihu.com/p/38163970

[4] Karush–Kuhn–Tucker conditions， Wikipedia

https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions

[5] 《綫性代數》，國立陽明交通大學，周志成博士

https://ccjou.wordpress.com/%E6%95%99%E5%AD%B8%E5%85%89%E7%A2%9F/

北京工业大学
BEIJING UNIVERSITY OF TECHNOLOGY

谢 谢!

不息为体
日新为道