Comparisons of Structure Learning Algorithms on Human Skeleton Data

Aodong Li
Computer Science Department
New York University
New York, NY 10003
al5350@nyu.edu

Abstract

Structure learning algorithms for graphical models vary over assumptions. Chow-Liu algorithm[3] assumes the model is a Bayesian network with tree structures; Graphical Lasso[5] encourages to seek sparse single-layer markov random fields. Researchers select suitable algorithms based on their understanding of the task properties. However, they seldom use them in a crossing way, although the structure that Chow-Liu algorithm learns is a subset of the ones learned by Graphical Lasso. In this project, we compare the algorithm performances when they are used in a less suitable scenario. In specific, we empirically study a case, modeling a human skeleton dataset UTD-MHAD[1], where Chow-Liu algorithm obviously takes a preference than Graphical Lasso. In terms of visual validity, we conclude that when we are solving a problem, at least for the graph structure learning, the match of the algorithm for the task assumption is more important than the potential of the algorithm. But we conserve the potentially existing opinion that the structure learned by Graphical Lasso conveys more information and may be useful for other applications.

1 Introduction

We consider the problem of using graphical structure learning algorithms in unsuitable situations. Structure learning algorithms usually learn specific kinds of graphical structure, by which the true probability distribution is modelled. The model structure is the our assumption of the task. In this project, we focus on two representative algorithms that learn different structures:

- Chow-Liu algorithm[3]: it learns a Bayesian network with tree structures by finding max spanning tree of mutual information using greedy algorithms.
- Graphical Lasso[5]: it estimates a sparse graph by exerting a lasso penalty on the inverse covariance matrix.

Graphical Lasso only makes sense for Gaussian graphical models (GGM) where each node of the graph is a Gaussian variable. Thus to gain a fair comparison, we also deploy Chow-Liu algorithm under the setting of GGMs.

We devise a scenario in learning the human skeleton structure where Chow-Liu algorithm takes a preference . And we compare the two algorithms' performances to see if the more powerful one – Graphical Lasso – can automatically capture the tree-structure assumption.

[1] http://www.utdallas.edu/ kehtar/UTD-MHAD.html

2 State-of-the-art

Many applications of human pose modeling concentrate in computer vision area to do pose estimation[6][2]. Some authors take it for granted that the skeleton follows from a tree-structure[1]. On the other hand, some authors argue that such a tree structure has obvious drawbacks like inability to incorporating gravity effect[7].

One work models the structure in non-parameteric setting[8] but does not learn a tree structure. They allege that the learned structure contains more meaningful information and is useful for downstream tasks like pose estimation.

Nevertheless, none of the work utilize Graphical Lasso on this task and compare the algorithm extensively.

3 Methods

To learn the structure, we utilize two algorithms.

3.1 Chow-Liu algorithm

For data distribution $p(\mathbf{x})$, we approximate it via a tree-structured Bayesian network $p_S(\mathbf{x})$ with a particular structure S. The tree-structure Bayesian networks define a family of product distributions where each node serves as one factor of at most second-order. For example, the distribution corresponding to the tree-structure directed acyclic graph (DAG) in Figure 1 is $p_S(\mathbf{x}) = p_S(x_1)p_S(x_2|x_1)p_S(x_3|x_1)$.

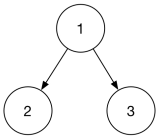

Figure 1: An example of DAG.

The goal of this second-order approximation is to select the most probable structure $S \in \mathcal{S}$. To measure this distance between $p(\mathbf{x})$ and $p_S(\mathbf{x})$, Chow-Liu algorithm[3] uses Kullback–Leibler (KL) divergence $D_{\text{KL}}(p(\mathbf{x})||p_S(\mathbf{x}))$. We can rephrase KL divergence in terms of entropy and mutual information in the following way,

$$
\begin{aligned}
D_{\text{KL}}(p(\mathbf{x})||p_S(\mathbf{x})) &= \int_{\mathbf{x}} p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{p_S(\mathbf{x})}\right) \mathrm{d}\mathbf{x} \\
&= \int_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) \mathrm{d}\mathbf{x} - \int_{\mathbf{x}} p(\mathbf{x}) \log \prod_i p_S(x_i|x_j(i)) \mathrm{d}\mathbf{x} \\
&= -\, h(\mathbf{x}) - \sum_i \int_{\mathbf{x}} p(\mathbf{x}) \log p_S(x_i|x_j(i)) \mathrm{d}\mathbf{x} \\
&= -\, h(\mathbf{x}) - \sum_i \int_{\mathbf{x}} p(\mathbf{x}) \log\left(\frac{p_S(x_i|x_j(i)) p_S(x_j(i))}{p_S(x_j(i)) p_S(x_i)}\right) \mathrm{d}\mathbf{x} \\
&\quad - \sum_i \int_{\mathbf{x}} p(\mathbf{x}) \log p_S(x_i) \mathrm{d}\mathbf{x} \\
&= -\, h(\mathbf{x}) - \sum_i \int_{x_i, x_j(i)} p(x_i, x_j(i)) \log\left(\frac{p_S(x_i, x_j(i))}{p_S(x_j(i)) p_S(x_i)}\right) \mathrm{d}x_i \mathrm{d}x_j(i) + \sum_i h(x_i) \\
&= -\, h(\mathbf{x}) - \sum_i I(x_i; x_j(i)) + \sum_i h(x_i)
\end{aligned}
$$

where $x_j(i)$ is the parent node of i induced by the graph structure S, and we assume the marginals of p ans p_S have the same density $p_S(x_i, x_j(i)) = \int_{\mathbf{x}_{\setminus x_i, x_j(i)}} p(\mathbf{x}) \mathrm{d}\mathbf{x}_{\setminus x_i, x_j(i)}$. $I(a; b) =$

$\int_{a,b} p(a,b) \log \frac{p(a,b)}{p(a)p(b)} \mathrm{d}a\mathrm{d}b$ is the mutual information of a and b. $h(a)$ is the differential entropy for continuous variables.

In order to minimize KL divergence between a data distribution and a distribution induced by a tree-structure Bayesian network, it is equivalent to find a particular tree structure that maximizes the sum of mutual information for each edge $\sum_i I(x_i; x_j(i))$. Thus a valid algorithm that finds such a structure can be summerized as below. First we compute mutual information between any two nodes. Second we build a max-spanning tree among the nodes. Then we add arrows from a source and propagate outward.

If we model the data distribution with multivariate normal distributions, then the mutual information becomes

$$I(x_i; x_j(i)) = -h((x_i, x_j(i))) + h(x_j(i)) + h(x_i).$$

For a k-dimensional multivariate normal distribution $p(\mathbf{x}) = \frac{\exp(-(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})/2)}{\sqrt{(2\pi)^k |\Sigma|}}$, the differential entropy is

$$\begin{aligned} h(\mathbf{x}) &= -\int_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &= \frac{1}{2} \mathrm{E}[(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})] + \frac{1}{2} \log |\Sigma| + \frac{k}{2} \log(2\pi) \\ &= \frac{k}{2} + \frac{1}{2} \log |\Sigma| + \frac{k}{2} \log(2\pi). \end{aligned}$$

Then the mutual information is

$$I(x_i; x_j(i)) = -\frac{1}{2} \log |\Sigma_{x_i,x_j(i)}| + \frac{1}{2} \log |\Sigma_{x_i}| + \frac{1}{2} \log |\Sigma_{x_j(i)}| = -\frac{1}{2} \log(|\Sigma_{x_i,x_j(i)}| / (|\Sigma_{x_i}||\Sigma_{x_j(i)}|)).$$

3.2 Graphical Lasso

Graphical Lasso algorithm[5] assumes that the observations are multivariate normal distribution. By inspecting the precision matrix we can determine the graphical model structure.

Recall multivariate normal distribution has its density

$$p(\mathbf{x}) = \frac{\exp(-(\mathbf{x}-\boldsymbol{\mu})^\top \Omega(\mathbf{x}-\boldsymbol{\mu})/2)}{\sqrt{(2\pi)^k |\Sigma|}}$$

where $\Omega = \Sigma^{-1}$ is the precision matrix. *Then variables i and j are conditionally independent given all the other variables if* $\Omega_{i,j} = 0$.

Because usually a lot of pairs of variables are conditionally independent, which corresponds to a sparse Markov random field, Graphical Lasso algorithm finds, for such a Markov random field, a sparse precision matrix by maximizing the log-likelihood

$$\begin{aligned} \mathcal{L}(\Omega) &= \frac{1}{n} \sum_{i=1}^{n} p(\mathbf{x}^{(i)}) = \frac{1}{2} \log |\Omega| - \frac{1}{2n} \sum_{i=1}^{n} \mathbf{x}^\top \Omega \mathbf{x} \\ &= \frac{1}{2} \log |\Omega| - \frac{1}{2} \langle \Omega, S \rangle = \frac{1}{2} \log |\Omega| - \frac{1}{2} \mathrm{tr}(S\Omega) \end{aligned}$$

where $S = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}\mathbf{x}^\top$ is the empirical covariance.

In order to encourage a sparse matrix Ω, an extra l_1 regularizer $\rho \|\Omega\|_1$ is added so that

$$\begin{aligned} &\text{maximize } \frac{1}{2} \log |\Omega| - \frac{1}{2} \mathrm{tr}(S\Omega) - \rho \|\Omega\|_1, \\ &\text{subject to } \Omega \succeq 0. \end{aligned}$$

We require $\Omega = \Sigma^{-1}$ to be positive semidefinite because if Σ is positive definite, Ω is also positive definite[2], and $\rho > 0$.

[2] Because their eigenvalues are reciprocal.

This problem is convex since the inner product and l_1 norm is convex and the operator $\log\det$ is concave with respect to positive semidefinite matrix. Thus we can solve it using CVX[3]. But a faster algorithm can be obtained by utilizing the first-order optimality condition

$$\Omega^{-1} - S - \rho\partial \|\Omega\| = 0,$$

for which we can find Ω^{-1} by a blockwise coordinate descent method[5].

4 Experiments

4.1 Skeleton modeling

For skeletons like human skeletons for which we focus on the skeleton joints, by assuming all the joints follow from multivariate normal distribution, we can model its distribution in space if we have a bunch of data showing different poses of the skeleton. Similar assumptions are also used in [4] and [8].

In three-dimensional space, we assume, for each skeleton joint, the coordinates (x, y, z) have a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance Σ, i.e., $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ or

$$p(\mathbf{x}) = \frac{\exp(-(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})/2)}{\sqrt{(2\pi)^k|\Sigma|}}$$

where $\mathbf{x} = (x, y, z)$.

4.2 Data

UTD-MHAD[4] is a dataset for human skeleton. Each sample has a set of 3D coordinates of a human skeleton with different actions. It has 58299 poses and each pose contains 20 joints. By taking our Gaussian distribution assumption, it means we have 58299 samples of 60-dimensional Gaussian random vector.

4.3 Results

4.3.1 Chow-Liu algorithm

To assign the edges, we use Kruskal's algorithm to maximize the sum of the mutual information.

The results of Chow-Liu algorithm are in Figure 2. It can be seen that only one edge – the connection between left knee and hip center – is incorrectly estimated.

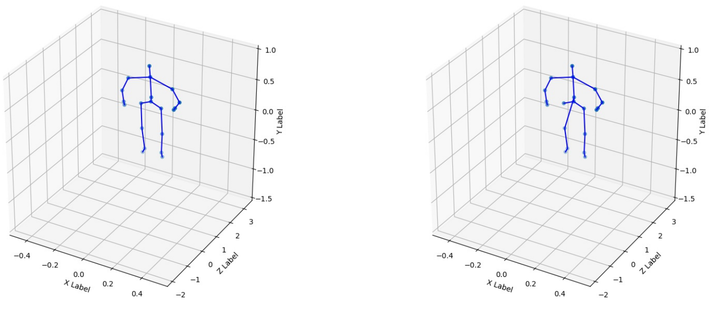

Figure 2: The left figure is for pulse-like signal and the right figure is for step-like signal.

[3] http://cvxr.com/cvx/

[4] http://www.utdallas.edu/ cxc123730/UTD-MHAD.html

4.3.2 Graphical Lasso algorithm

We use the solver from Scikit-learn[9] package.

Since each node is a three-dimensional multivariate Gaussian variable, the learned precision matrix is 60×60. To assign the edges from this precision matrix, we assign an edge between two nodes at least three entries in their 3×3 sub-matrix are nonzero.

The results of Graphical Lasso algorithms are in Figure 3. It can be seen that as we increase the penalty paramter ρ, the structure becomes sparser, but the it is not better than Chow-Liu algorithm.

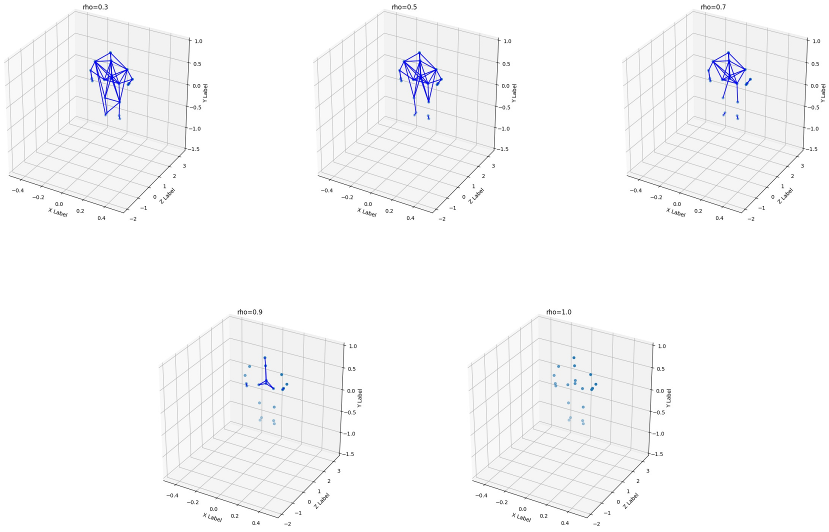

Figure 3: From left to right are the learned structures for regularizer weight $\rho = 0.3, 0.5, 0.7, 0.9, 1.0$.

5 Discussion

5.1 Conclusion

In terms of visual validity, when dealing with a problem, at least for graphical model structure learning, instead of choosing a powerful one, an algorithm with matched assumption is more important.

However, we have not compared the log-likelihood. A potential consequence is that the structure learned by Graphical Lasso actually conserves more information and it is the assumption of Chow-Liu algorithm that limits its ability to capture that part of information. This information may be useful for other applications and can be seen as a prior for them.

5.2 Future work

- For this task, blockwise Graphical Lasso should be more appropriate because each joint consists of three variables, such an algorithm should be further investigated.
- The way of assigning the edges between two multivaraite vectors should also be investigated.
- The log-likelihood for the two algorithms should also be compared to see which one contains more information. To explore which applications might be absorbed to this information is another interesting field.

References

[1] Christoph Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pages 8–15. IEEE, 1998.

[2] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 479–488. IEEE, 2016.

[3] C Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[4] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[6] Fei Han, Brian Reily, William Hoff, and Hao Zhang. Space-time representation of people based on 3d skeletal data: A review. *Computer Vision and Image Understanding*, 158:85–105, 2017.

[7] Xiangyang Lan and Daniel P Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 470–477. IEEE, 2005.

[8] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. A non-parametric bayesian network prior of human pose. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1281–1288, 2013.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.