208891 Probabilistic Graphical Models **Spring 2019**

Lecture 10: February 4

Lecturer: Donlapark Pornnopparath

10.1 Structure search of general graphs

To make computation easier to handle, we consider the family of graph $\mathcal{G}_d = \{G \in \mathcal{G} | \text{for each variable } X_i, |\text{Pa}(X_i)| \leq d\}$. However, even with the restriction, the problem of finding the network that maximizes the score is still complicated as it is NP-hard for any $d \geq 2$, meaning that there is no guarantee (at least for now) of an algorithm that can solve this problem in polynomial time. Thus, we have to come up with a heuristic approach that tries to find a graph with high score but does not guarantee the maximum. The typical search procedure consists of the following components.

Initialization We can be flexible in how we start with the first graph as this would help us expanding the search space. Typically, the initial graph is one of the following:

- an empty graph;
- a random graph;
- the maximum weight spanning forest.

Local search In each step, we apply a local change to the graph using one of the following operations:

- edge addition;
- edge deletion;
- edge reversal.

The decomposability of the score allows the computation of the algorithm to be feasible; if one of the operations f is applied to the graph G to obtain $f(G)$, then the difference of the score is

$$\begin{aligned}\Delta G &= s(f(G)|D) - s(G|D)\\ &= \begin{cases} \text{Score}(Y|\text{Pa}(Y) \cup \{X\}, D) - \text{Score}(Y|\text{Pa}(Y), D) & f \text{ is ``Add } X \to Y\text{''}.\\ \text{Score}(Y|\text{Pa}(Y) - \{X\}, D) - \text{Score}(Y|\text{Pa}(Y), D) & f \text{ is ``Delete } X \to Y\text{''}.\\ \text{Score}(X|\text{Pa}(X) \cup \{Y\}, D) + \text{Score}(Y|\text{Pa}(Y) - \{X\}, D) & \\ \quad -\text{Score}(X|\text{Pa}(X), D) - \text{Score}(Y|\text{Pa}(Y), D) & f \text{ is ``Reverse } X \to Y\text{''}.\end{cases}\end{aligned}$$

Thus, we do not have to recompute the sufficient statistics of the whole graph, only at the nodes that were recently changed. We can also save these computations in the cache and reuse it again when applying the same operation to a different graph G' which has the same local structure at Y i.e. same set of $\text{Pa}(Y)$.

There are several ways to apply these operations.

1. **Hill-climbing** The most basic search algorithm. In each step, all possible edge additions, deletions and edge reversals are considered and the one that maximizes the change of the score is chosen. The algorithm stops when performing any of the available operations will not improve the score. The problem with this method is that the solution usually gets stuck in a local, but not global, maxima.

2. **Basin-flooding** This is similar to hill-climbing, but instead of disregarding the possible alternatives in each step, we take the record of the previous graphs. After reaching a local maxima, we continue to find an operation that leads to a different answer that the ones we recorded. This method allows us to escape from a local maxima but is very memory and computational intensive.

3. **tabu search** This is a modification of previous two algorithms. The method records all the operations in the previous L (a specified number) steps. To modify the current network, we do not consider any operation that reverses the effect of the recorded operations. For example, if there has been an addition of $X \to Y$ in the previous L steps, then we cannot remove the $X \to Y$ edge in the next step. After the algorithm reaches a local maxima, the searching is still continue until a specified number of steps is reached. This method is not as computational expensive as basin-flooding and allows us to explore in new directions beyond the local maxima.

4. **Data perturbation** This method is different than the previous ones in that we modify the data instead of the searching procedure. Basic data perturbations include instance duplication and removal. By applying these modifications to some of the instances, we obtain a new dataset D' which has similar characteristics as the original dataset D. In particular, their sufficient statistics should be roughly the same, and the shape of the score function should be preserved on the global scale. In summary, we keep exploring for new solutions by perturbing D, doing a local search on the new dataset, and then check if the solution is an actual improvement of the previous one. An alternative way of modifying the data is by increasing the number of instances by a factor of w: $x^{(i)} \to x^{(i)} \cdot w$, where w is a positive random variable with mean 1 and variance t, usually sampled from a Gamma distribution.

Another technique that help expanding the search space is the *random restarts* where a new search is initialized with a random graph after the previous algorithm is terminated.