Structured Prediction of Network Response
Supplementary Material

Hongyu Su HONGYU.SU@AALTO.FI

Aristides Gionis ARISTIDES.GIONIS@AALTO.FI

Juho Rousu JUHO.ROUSU@AALTO.FI

Helsinki Institute for Information Technology (HIIT)
Department of Information and Computer Science, Aalto University, Finland

1. NP-hardness of the inference problem

Lemma 1 *Finding the graph that maximizes Eq.(2) is an* **NP***-hard problem.*

Proof (Sketch) The inference problem of SPIN can be stated as searching for an optimal configuration of node labels. The node labels $\{y_v \in \{\mathtt{p},\mathtt{n}\}\}_{v\in V}$ induce edge labels $\{y_e \in \{\mathtt{nn},\mathtt{pn},\mathtt{pp}\}\}_{e\in E}$. For each edge e and its label y_e in the graph, a score $s_{y_e}(e)$ is defined. The goal is to find a set of node labels that maximizes the sum of scores $\sum_{e\in E} s_{y_e}(e)$.

This inference problem can be seen as a generalization of the MAX-CUT problem (Garey & Johnson, 1979), where the objective is to partition the nodes of a given undirected graph into two parts so that the number of edges cut is maximized. The MAX-CUT problem is known to be **NP**-hard.

We give a reduction from the MAX-CUT problem to the inference problem. Given an instance of MAX-CUT, i.e., a undirected graph $G = (V, E)$, we create an instance to our problem as follows. We first create a directed graph $G' = (V, E')$ by considering both directions of the edges of G, i.e., for each $(u, v) \in E$ we add (u, v) and (v, u) in E'. Then, for each directed edge $e \in E'$ we set the scores $s_{\mathtt{pp}}(e) = s_{\mathtt{nn}}(e) = 0$ and $s_{\mathtt{pn}}(e) = 1$. In addition, the two partitions in MAX-CUT are represented via the label $\{\mathtt{p},\mathtt{n}\}$. It is not difficult to see that the reduction only requires linear time and space, and an optimal solution to the MAX-CUT problem is also optimal to this special case of the inference problem. □

2. Derivation and pseudo-code of GREEDY inference algortihm

The original inference problem is stated as maximizing the sum over the scores of consistent edge labels. To tackle the inference problem, one has to work with all edges and labels. We show in Eq. 1 that the inference problem can be equivalently expressed only in term of activated vertices and the associated scores by initializing the search from a negative network. The alternative expression of the inference problem enables us to design an iterative greedy search algorithm that works on the nodes of the network.

Pseudocode for greedy inference algorithm is shown in Algorithm 1. The algorithm starts with an activated vertex set $V_{\mathtt{p}}^H = \{r\}$ that only contains root node r (line 1). It also maintains a priority list of vertices by their current scores $F_m(v)$ (line 2). In each iteration (line 4), the algorithm pops from the top of the priority list the vertex v_i that is the current maximizer of the marginal gain

$$v_i = \underset{v_i \in V/V_{\mathtt{p}}^H}{\textbf{argmax}}\ F_m(v_i).$$

Then, for all neighbours of v_i, the algorithm updates their scores $F_m(v)$ and positions in the list (line 6). The procedure terminates if $F_m(v_i) < 0$ (line 7). E^H can be obtained by adding edges $e = (v_i, v_j) \in E$, if $v_i, v_j \in V_{\mathtt{p}}^H$ and v_i was added to $V_{\mathtt{p}}^H$ prior to v_j (line 13).

It is easy to see that it takes at most $O(\log |V|)$ to update the score and the position of one child node of current activated node (line 6). In addition, we notice that each update corresponds to an edge in the network. Once the edge is used for update, it will never be used again. Thus, the complexity for greedy search is $O(|E| \log |V|)$.

$$
\begin{aligned}
H^*(\mathbf{a}) &= \underset{H \in \mathcal{H}(G)}{\textbf{argmax}} \underbrace{\sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{p}}^H}} s_{pp}(v_i,v_j)}_{activated\,network} + \underbrace{\sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{n}}^H}} s_{pn}(v_i,v_j)}_{boundary} + \underbrace{\sum_{\substack{v_i \in V_{\mathrm{n}}^H \\ v_j \in V_{\mathrm{n}}^H}} s_{nn}(v_i,v_j)}_{inactivated\,network} \qquad (1) \\
&= \underset{H \in \mathcal{H}(G)}{\textbf{argmax}} \sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{p}}^H}} s_{pp}(v_i,v_j) + \sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{n}}^H}} s_{pn}(v_i,v_j) + \sum_{\substack{v_i \in V_{\mathrm{n}}^H \\ v_j \in V_{\mathrm{n}}^H}} s_{nn}(v_i,v_j) - \sum_{\substack{v_i \in V^H \\ v_j \in V^H}} s_{nn}(v_i,v_j) \\
&= \underset{H \in \mathcal{H}(G)}{\textbf{argmax}} \sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{p}}^H}} [s_{pp}(v_i,v_j) - s_{nn}(v_i,v_j)] + \sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{n}}^H}} [s_{pn}(v_i,v_j) - s_{nn}(v_i,v_j)] \\
&= \underset{H \in \mathcal{H}(G)}{\textbf{argmax}} \underbrace{\sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{p}}^H}} [s_{pp}(v_i,v_j) - s_{pn}(v_i,v_j)] + \sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{p}}^H}} [s_{pn}(v_i,v_j) - s_{nn}(v_i,v_j)]}_{activated\,network} + \underbrace{\sum_{\substack{v_i \in V_{\mathrm{p}}^H \\ v_j \in V_{\mathrm{n}}^H}} [s_{pn}(v_i,v_j) - s_{nn}(v_i,v_j)]}_{boundary} \\
&= \underset{H \in \mathcal{H}(G)}{\textbf{argmax}} \sum_{v_i \in V_{\mathrm{p}}^H} \left(\underbrace{\sum_{v_p \in pa(v_i)} [s_{pp}(v_p,v_i) - s_{pn}(v_p,v_i)]}_{incoming\,edges} + \underbrace{\sum_{v_c \in chi(v_i)} [s_{pn}(v_i,v_c) - s_{nn}(v_i,v_c)]}_{outgoing\,edges} \right) \\
&= \underset{H \in \mathcal{H}(G)}{\textbf{argmax}} \sum_{v_i \in V_{\mathrm{p}}^H} F_m(v_i).
\end{aligned}
$$

References

Garey, Michael R. and Johnson, David S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

Algorithm 1 $\mathrm{Greedy\_Inference}(a, G, s_{\mathbf{u}_e}(e))$

Input: Action $\mathbf{a}$. Network $G = (E, V)$. The score function $s_{\mathbf{u}_e}(e)$ defined on edge e and label $\mathbf{u}_e$.
Output: A rooted DAG $H = (V^H, E^H)$ that maximizes the score function

$$H^*(\mathbf{a}) = \mathbf{argmax}_{H \in \mathcal{H}(G)} \sum_{v_i \in V_{\mathrm{p}}^H} F_m(v_i)$$

$V_{\mathrm{p}}^H = \{r\}$, $E^H = \{\}$, $T(r) = 0$, $t = 1$
Initialize a priority list $V' = \{v | v \in V\}$ ordered by $F_m(v)$.
while true do
  Pop up the first element from V' such that

$$v = \mathbf{argmax}_{v \in V / V_{\mathrm{p}}^H} F_m(v)$$

  $V' = V' - \{v\}$
  Updated scores $F_m(v)$ and positions for all neighbours of v.
  if $F_m(v) \geq 0$ **then**
    $V_{\mathrm{p}}^H = V_{\mathrm{p}}^H \cup \{v\}$
    $T(v) = t$
    $t = t + 1$
  else
    break
for $v_i \in V_{\mathrm{p}}^H, v_j \in V_{\mathrm{p}}^H$ **do**
  if $(v_i, v_j) \in E$ **and** $T(v_i) < T(v_j)$ **then**
    $E^H = E^H \cup \{(v_i, v_j)\}$
$H = (E^H, V^H)$