Distribution Learning Meets Graph Structure Sampling

Sayantan Sen

Centre for Quantum Technologies

NUS National University of Singapore

Arnab Bhattacharyya

NUS National University of Singapore

Sutanu Gayen

Indian Institute of Technology Kanpur

Philips George John

NUS National University of Singapore

N. V. Vinodchandran

University of Nebraska Lincoln

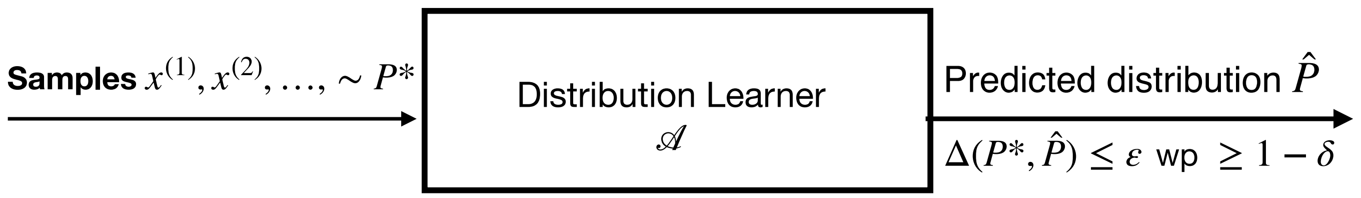
WOLA 2024

Plan of the talk

- Distribution Learning
- Online Learning
- Interplay of Distribution and Online Learning
- Summary of our results
- Overview of our technical framework

Distribution Learning

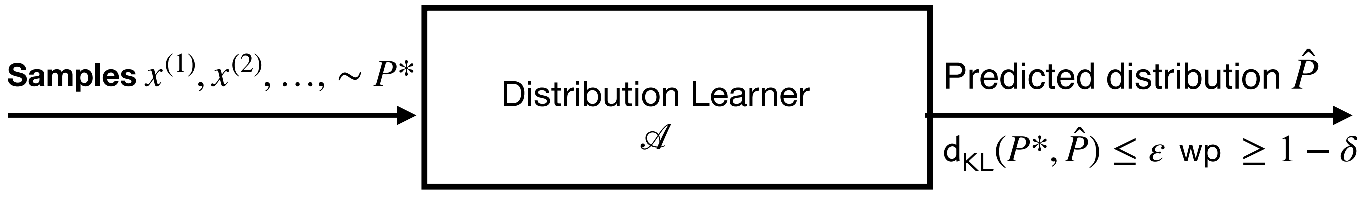
Given samples from an unknown distribution P^*, we want to “learn” a distribution $\hat{P}$ which is “close to” P^*.


Samples $x^{(1)}, x^{(2)}, \ldots, \sim P^*$
Distribution Learner
$\mathscr{A}$
Predicted distribution $\hat{P}$
$\Delta(P^*, \hat{P}) \leq \varepsilon$ wp $\geq 1 - \delta$

- $\hat{P}$ should generally be “efficiently sampleable” (can return a sampler).
- May want $\hat{P}$ to have a specific structure. Assumptions (if any) about P^*.

$$\mathrm{d}_{\mathrm{KL}}(P^* \quad \hat{P}) = \sum_x P^*(x) \log \frac{P^*(x)}{\hat{P}(x)}$$

Distribution Learning Contd.



Samples $x^{(1)}, x^{(2)}, \ldots, \sim P^*$
Distribution Learner
$\mathscr{A}$
Predicted distribution $\hat{P}$
$d_{KL}(P^*, \hat{P}) \leq \varepsilon$ wp $\geq 1 - \delta$

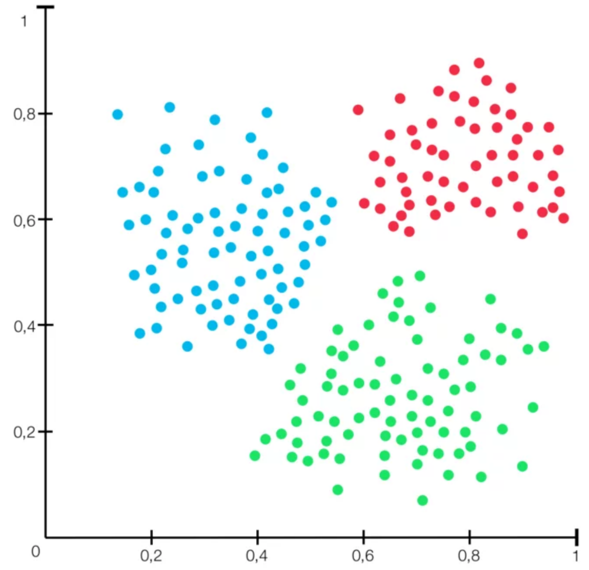
- Pertinent complexity measures for a distribution learning algorithm include:
- Sample complexity (# of samples needed for theoretical guarantees to hold)
- Time complexity (running time)

Algorithm design might involve **trade-offs** between these two factors.

Distribution Learning: Motivation

A lot of machine learning is implicitly distribution learning where the learnt distribution is on $\mathscr{X}$, given by the data/features distribution (over $\mathscr{X}$ and the classification or regression model $f : \mathscr{X} \to \mathscr{Y}$.

2D clustering problem



Induces a distribution on

$$\underbrace{[0,1]^2}_{\text{data}} \times \underbrace{\{0,1,2\}}_{\text{labels}}.$$

Learning this distribution gives a clustering method (after marginalization, choose the marginal with maximum likelihood).

Learning High Dimensional Distributions

- Distribution learning is non-trivial even with discrete distributions, when the *domain* is large, e.g $[k]^n$.
- Takes exponential number of samples and time in general.

$\Omega(k^n)$ samples are required for learning *arbitrary distributions* over $[k]^n$.

Many use cases involve high dimensional distributions:

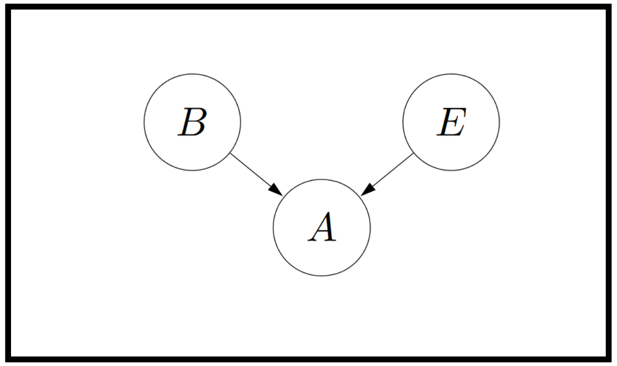
- Machine Learning
- Program Analysis

Can we learn important subclasses?

Bayesian Networks

A distribution P over n variables $X_1, \dots, X_n$ is a Bayesian Network on a DAG $G = ([n], E)$ if P factories as follows:

$$P(x) = \prod_{i=1}^{n} \Pr_{X \sim P}\left(X_i = x_i \mid \forall j \in \mathrm{pa}(i), X_j = x_j\right)$$



$$P(A = a, B = b, E = e) = p_B(b) \cdot p_E(e) \cdot p_A(a \mid b, e)$$

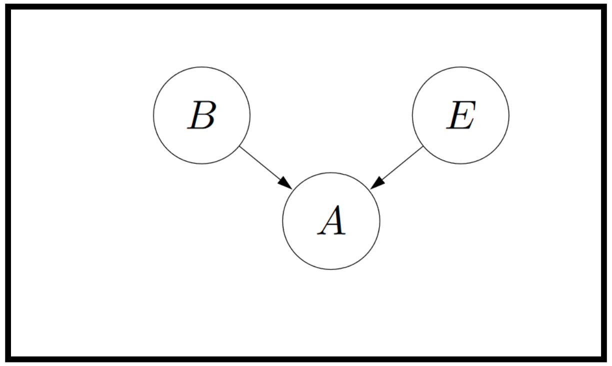
A Bayes net distribution P can be represented by $\left(G, P = \{p_i\left(X_i \mid X_{pa(i)}\right)\}_{i \in [n]}\right)$.

Representation requires $O(nk^{d+1})$ space.

Bayes Nets Contd.

Indegree, treewidth etc. of a Bayes net refers to the indegree, treewidth etc. of G.

(in)degree of G = maximum (in)degree of its vertices.

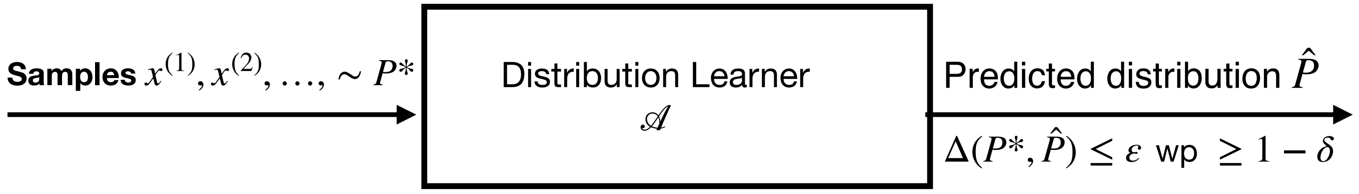


Any distribution $P(X_1, \ldots, X_n)$ can be represented by a Bayes net with indegree $\leq n - 1$.

$$P(A = a, B = b, E = e) = p_B(b) \cdot p_E(e) \cdot p_A(a \mid b, e)$$

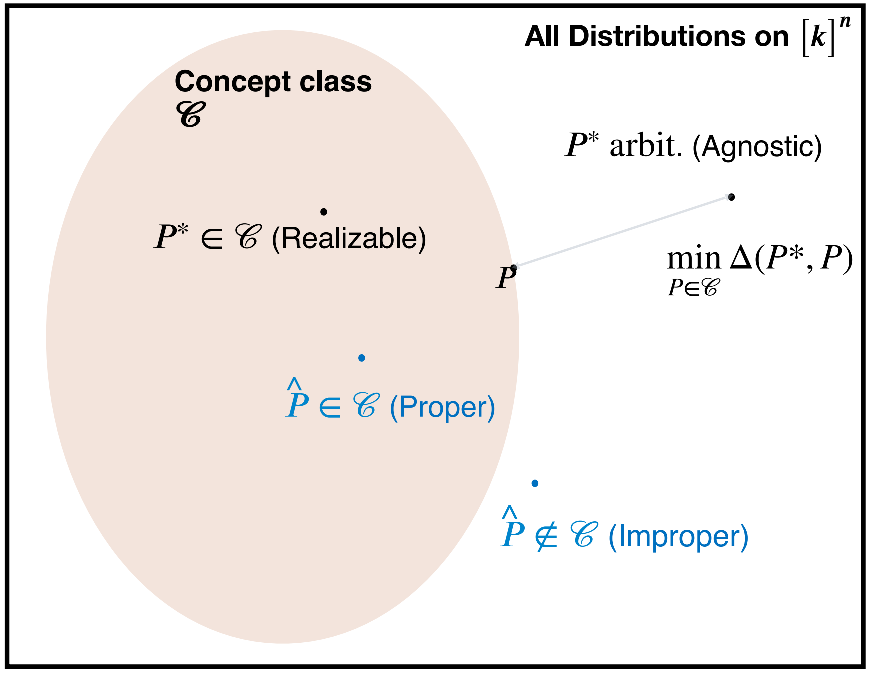
$$P(x) = \prod_{i=1}^{n} \Pr_{X \sim P} \left(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}\right)$$

Distribution Learning Variants



Samples $x^{(1)}, x^{(2)}, \ldots, \sim P^*$
Distribution Learner $\mathscr{A}$
Predicted distribution $\hat{P}$
$\Delta(P^*, \hat{P}) \leq \varepsilon$ wp $\geq 1 - \delta$

- Realizable Learning: $P^* \in \mathscr{C}$
- Agnostic Learning: P^* arbitrary
- Proper Learning: $\hat{P} \in \mathscr{C}$
- Improper Learning: $\hat{P}$ is mixture of $\mathscr{C}$ distributions



All Distributions on $[k]^n$
Concept class $\mathscr{C}$
P^* arbit. (Agnostic)
$P^* \in \mathscr{C}$ (Realizable)
P
$\min_{P \in \mathscr{C}} \Delta(P^*, P)$
$\hat{P} \in \mathscr{C}$ (Proper)
$\hat{P} \notin \mathscr{C}$ (Improper)

Our results: Spotlight 1

Efficient learning algorithm for **tree-structured distributions** that requires $\widetilde{O}\left(\frac{n\mathbf{k}^{\mathbf{2}}}{\varepsilon}\right)$ and $\widetilde{O}\left(\frac{n^4k^4}{\varepsilon^4}\right)$ samples respectively in the realizable and agnostic cases.

- First efficient algorithm that does not use the Chow-Liu approach.

- Sample complexity better in terms of k (k^2 vs k^3) compared to the Chow-Liu approach in the realizable case.

Our results: Spotlight 2

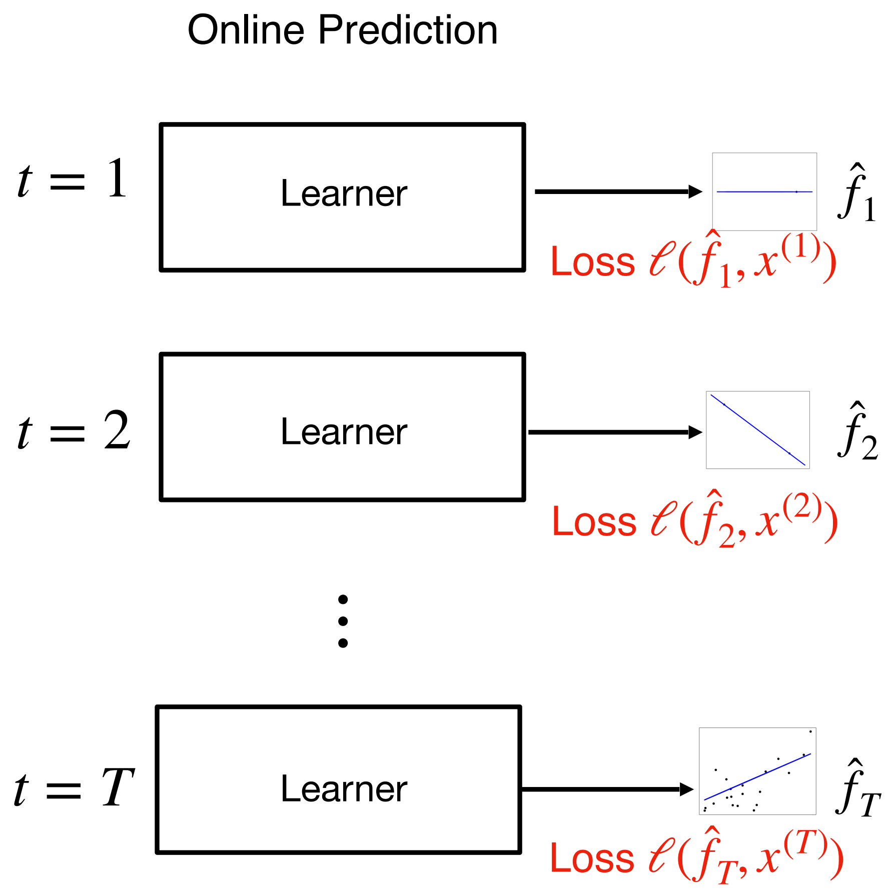
First **efficient** algorithm for learning **chordal-structured distributions when the skeleton is known**. Sample complexity $\tilde{O}\left(\frac{n^3k^{d+1}}{\varepsilon^2}\right)$.

- Chordal-structured distributions form a large and interesting class of Bayes nets over $[k]^n$.
- Covers tree-structured distributions, polytree-structured distributions etc.
- Previously, no efficient algorithms for learning even if skeleton was known.

Online Learning

Online Learning (Prediction)

- Before seeing outcome $x^{(t)} \in \mathscr{X}$, learner predicts $\hat{f}_t \in \mathscr{D}$.
- After prediction, learner sees $x^{(t)}$ and suffers loss $\ell(\hat{f}_t, x^{(t)})$.
- $x^{(1)}, \ldots, x^{(T)}$ can be arbitrary.


Online Prediction
$t = 1$
Learner
$\hat{f}_1$
Loss $\ell(\hat{f}_1, x^{(1)})$
$t = 2$
Learner
$\hat{f}_2$
Loss $\ell(\hat{f}_2, x^{(2)})$
$t = T$
Learner
$\hat{f}_T$
Loss $\ell(\hat{f}_T, x^{(T)})$

Prediction with Expert Advice

- The online learning algorithm $\mathscr{A}$ is given a set of experts $\mathscr{E} = \{E_1, \ldots, E_N\}$.
- Suppose each E_i corresponds to a prediction in $\mathscr{D}$.
- $\mathscr{A}$ predicts $\hat{f}_t$ based on $\{E_1, \ldots, E_N\}$ before seeing $x^{(t)}$.
- $\mathscr{A}$ suffers loss $\ell(\hat{f}_t, x^{(t)})$. E_i suffers loss $\ell(E_i, x^{(t)})$.

Regret of learner $\mathscr{A}$ wrt $\mathscr{E}$ is:

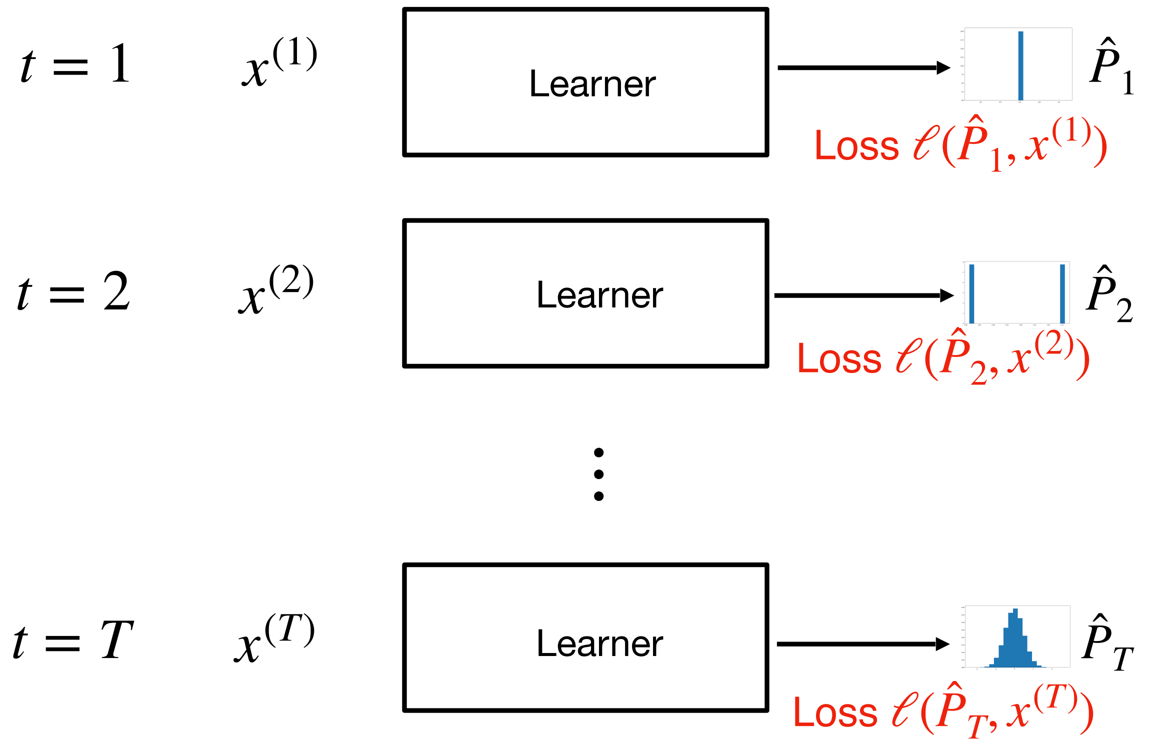
$$Reg_T(\mathscr{A}; \mathscr{E}) = \underbrace{\sum_{t=1}^{T} \ell(\hat{P}_t, x^{(t)})}_{\text{Total loss of } \mathscr{A}} - \underbrace{\min_{E \in \mathscr{E}} \sum_{t=1}^{T} \ell(E, x^{(t)})}_{\text{Total loss of bext expert}}$$

Online Distribution Learning

- Before seeing $x^{(t)}$, learner predicts $\hat{P}_t$.

- After prediction, learner sees $x^{(t)}$ and suffers loss $\ell(\hat{P}_t, x^{(t)})$.

- $x^{(1)}, \ldots, x^{(T)}$ are iid samples.



Every Expert will be a candidate distribution!

Online Distribution Learning Contd.

Regret of an online distribution learning algorithm $\mathscr{A}$ wrt class of distributions $\mathscr{C}$ is

$$Reg_T(\mathscr{A};\mathscr{C}) = \underbrace{\sum_{t=1}^{T} \ell(\hat{P}_t, x^{(t)})}_{\text{Total loss of } \mathscr{A}} - \underbrace{\min_{E \in \mathscr{C}} \sum_{t=1}^{T} \ell(P, x^{(t)})}_{\text{Total loss of bext expert}}$$

- Useful to have algorithms with $Reg_T(\mathscr{A}, \mathscr{C}) = o(T)$.
- Average regret $\dfrac{Reg(\mathscr{A}, \mathscr{C})}{T} = o(1)$.

No regret learning!

Interplay of Distribution Learning & Online Learning

For $\ell(P, x)$ is log loss and $x^{(1)}, \ldots, x^{(T)} \sim P^*$,

$$\boxed{\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}\sim P^*} \mathop{\mathbb{E}}_{t\sim\mathsf{Unif}([T])} \left[\mathsf{d}_{\mathsf{KL}}(P^* \| \hat{P}_t)\right] \leq \frac{1}{T} \mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}} \left[Reg_T(\mathscr{A}\,;\,\mathscr{C})\right] + \min_{P\in\mathscr{C}} \mathsf{d}_{\mathsf{KL}}(P^* \quad P)}$$

Low regret online algorithms gives distribution learning algorithms!

Interplay of Distribution Learning & Online Learning Contd.

$$\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}\sim P^*} \mathop{\mathbb{E}}_{t\sim \mathsf{Unif}([T])} \left[\mathsf{d}_{\mathsf{KL}}(P^* \| \hat{P}_t)\right] \leq \frac{1}{T} \mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}} \left[Reg_T(\mathscr{A}\,;\,\mathscr{C})\right] + \min_{P\in\mathscr{C}} \mathsf{d}_{\mathsf{KL}}(P^* \quad P)$$

- If we run $\mathscr{A}$ for large enough T, then

$$\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}} \left[\mathsf{d}_{\mathsf{KL}}\left(P^* \| \frac{1}{T}\sum_{t=1}^{T} \hat{P}_t\right)\right] \leq \min_{P\in\mathscr{C}} \mathsf{d}_{\mathsf{KL}}(P^* \quad P) + \varepsilon$$

- Apply concentration bounds for high probability guarantee.

Proof Sketch of Reg-AL Lemma

Lemma: $$\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}\sim P^*}\ \mathop{\mathbb{E}}_{t\sim\mathsf{Unif}([T])}\left[\mathsf{d}_{\mathsf{KL}}(P^*\|\hat{P}_t)\right] \le \frac{1}{T}\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}}\left[Reg_T(\mathscr{A}\,;\,\mathscr{C})\right] + \min_{P\in\mathscr{C}} \mathsf{d}_{\mathsf{KL}}(P^* \quad P)$$

$$\frac{1}{T}Reg_T(\mathscr{A},\mathscr{C}) = \frac{1}{T}\sum_{t=1}^{T}\log\frac{1}{\hat{P}_t(x^{(t)})} - \frac{1}{T}\min_{P\in\mathscr{C}}\sum_{t=1}^{T}\log\frac{1}{P(x^{(t)})}$$

$$= \frac{1}{T}\sum_{t=1}^{T}\log\frac{P^*(x^{(t)})}{\hat{P}_t(x^{(t)})} - \min_{P\in\mathscr{C}}\frac{1}{T}\sum_{t=1}^{T}\log\frac{P^*(x^{(t)})}{P(x^{(t)})}$$

Linearity of expectation & law of conditional expectation

$$\frac{1}{T}\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}} Reg_T(\mathscr{A},\mathscr{C}) = \frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(t-1)}}\ \mathop{\mathbb{E}}_{x^{(t)}\sim P^*}\left[\log\frac{P^*(x^{(t)})}{\hat{P}_t(x^{(t)})}\,\middle|\,x^{(1)},\ldots,x^{(t-1)}\right] - \mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}}\min_{P\in\mathscr{C}}\frac{1}{T}\sum_{t=1}^{T}\left[\log\frac{P^*(x^{(t)})}{P_t(x^{(t)})}\right]$$

$$\ge \frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(t-1)}}\ \mathop{\mathbb{E}}_{x^{(t)}\sim P^*}\left[\log\frac{P^*(x^{(t)})}{\hat{P}_t(x^{(t)})}\,\middle|\,x^{(1)},\ldots,x^{(t-1)}\right] - \min_{P\in\mathscr{C}}\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{x^{(t)}}\left[\log\frac{P^*(x^{(t)})}{P(x^{(t)})}\right]$$

Jensen's ineq. and linearity of expectation

$$= \mathop{\mathbb{E}}_{x^{(1)},\ldots,x^{(T)}}\ \mathop{\mathbb{E}}_{t\sim\mathsf{Unif}([T])}\mathsf{d}_{\mathsf{KL}}\left(P^* \quad \hat{P}_t\right) - \min_{P\in\mathscr{C}}\mathsf{d}_{\mathsf{KL}}\left(P^* \quad P\right)$$

Rearranging gives the lemma!

Our results

Learning Bayes nets Bounds

$\mathscr{C}$ = { Bayes nets of indegree $\leq d$ over $[k]^n$ }

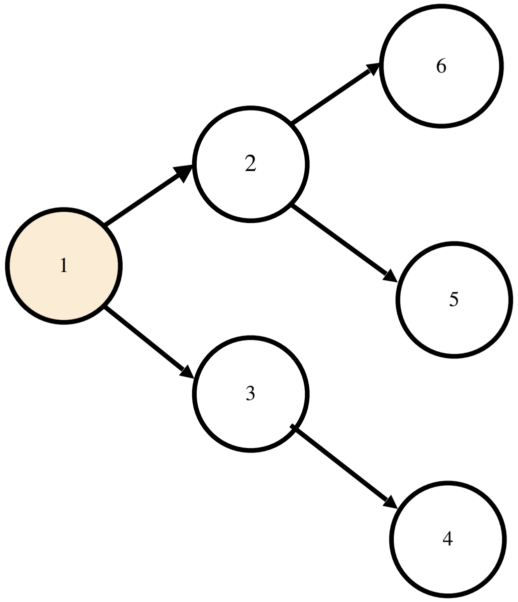
These algorithms are time inefficient!

Sample Complexity	Realizable	Agnostic
Improper Learning	$\tilde{O}\left(\frac{nk^{d+1}}{\varepsilon}\log\frac{1}{\delta}\right)$	$\tilde{O}\left(\frac{n^4k^{2d+2}}{\varepsilon^4}\log\frac{1}{\delta}\right)$
Proper Learning	$\tilde{O}\left(\frac{n^2k^{d+1}}{\varepsilon^2}\right)$ (BGPTV21)	$\tilde{O}\left(\frac{n^3k^{d+1}}{\varepsilon^2\delta^2}\right)$
Lower bound	$\Omega\left(\frac{nk^{d+1}}{\varepsilon}\right)$ (BCD20)	

Tree-structured Distributions

Let T be a tree on $[n]$, and G_T be any rooted orientation of T aka *out-arborescence* (all edges directed outwards from a fixed root node).

A distribution P is tree-structured (aka a tree Bayes net) if it is a Bayes net on G_T for some tree T.



$$P(x) = p_1(x_1) \cdot p_2(x_2 \mid x_1) \cdot p_3(x_3 \mid x_1) \cdot p_4(x_4 \mid x_3) \cdot p_5(x_5 \mid x_2) \cdot p_6(x_6 \mid x_2)$$

Our results for Tree-struct. Distributions

Runs in Poly$(n, \frac{1}{\varepsilon}, \frac{1}{\delta})$ time!

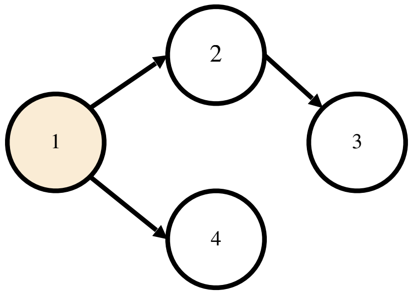
Sample Complexity	Realizable	Agnostic
Improper Learning	$\tilde{O}\left(\frac{nk^2}{\varepsilon\delta}\right)$ Improves the dependency of k^2	$\tilde{O}\left(\frac{n^4k^4}{\varepsilon^4}\log\frac{1}{\delta}\right)$
Proper Learning	$\tilde{O}\left(\frac{nk^3}{\varepsilon}\right)$	$\tilde{O}\left(\frac{n^3k^2}{\varepsilon^2\delta^2}\right)$
Lower bound	$\Omega\left(\frac{nk^2}{\varepsilon}\right)$(CDKS17)	$\Omega\left(\frac{n^2}{\varepsilon^2}\right)$ (BGPTV21, DP21)

Chordal-structured Distributions

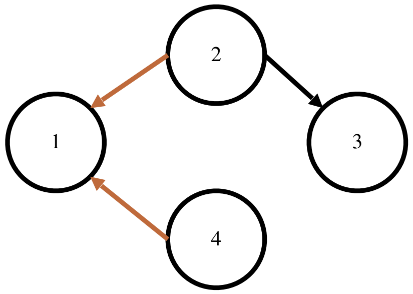
G = undirected *chordal graph (*all cycles of length ≥ 4 have *chord edges*)*.*

$\bar{G}$ = any DAG with *skeleton* (underlying undirected graph) G.

Distribution P is **chordal-structured with skeleton G** if it is a Bayes net on $\bar{G}$.

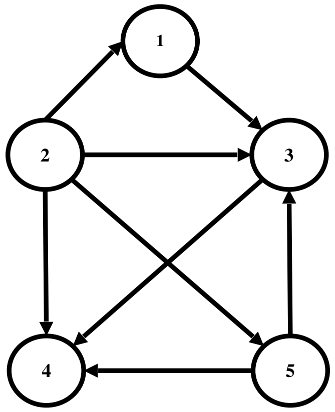


Tree-structured



Polytree-structured

(tree skeleton, DAG oriented arbitrarily)



Chordal-structured

(non-tree skeleton)

Our results for Chordal-structured distribution

G is undirected chordal graph.

$\mathscr{C}$ = { Bayes nets with skeleton G of indegree $\leq d$ over $[k]^n$}

Sample Complexity	Agnostic
Improper Learning	$\tilde{O}\left(\frac{n^4 k^{2d+2}}{\varepsilon^4} \log \frac{1}{\delta}\right)$
Proper Learning	$\tilde{O}\left(\frac{n^3 k^{d+1}}{\varepsilon^2 \delta^2}\right)$
Lower bound	$\Omega\left(\frac{n k^{d+1}}{\varepsilon}\right)$ (CYBC24)

Runs in Poly$(n, \frac{1}{\varepsilon}, \frac{1}{\delta})$ time!

Our Techniques

Our Algorithm Framework 1: Discreatization

For a class $\mathscr{C}$ of Bayes nets, discreatize distributions in $\mathscr{C}$ to a finite set $\mathscr{N} \subset C$ such that

- Clipping: $-\log P(x)$ is upper bounded for all $P \in \mathscr{N}$.

- Bucketing: Bound regret wrt $\mathscr{N}$ close to regret wrt $\mathscr{C}$.

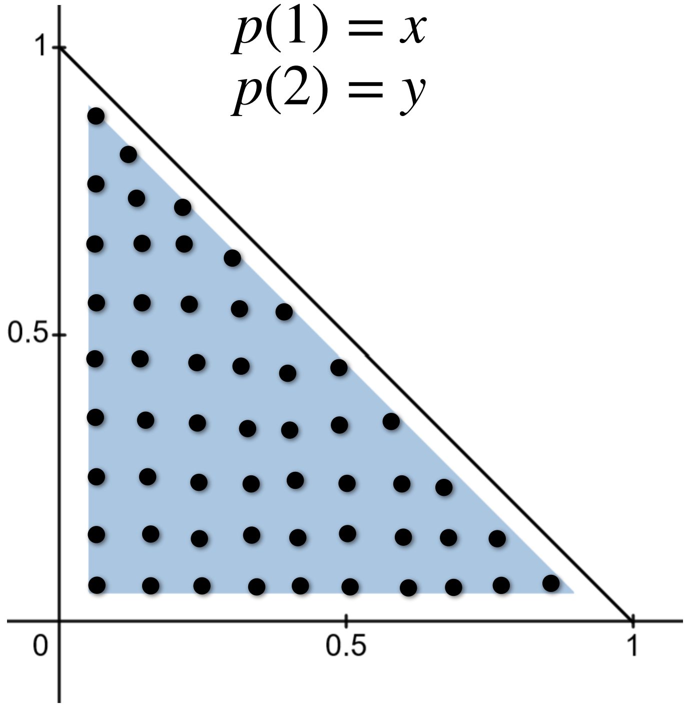
Distribution over $\{0,1,2\}$

$p = (x, y)$ in outer triangle

$p(0) = 1 - x - y$

$p(1) = x$

$p(2) = y$



Our Algorithm Framework 2: Learning

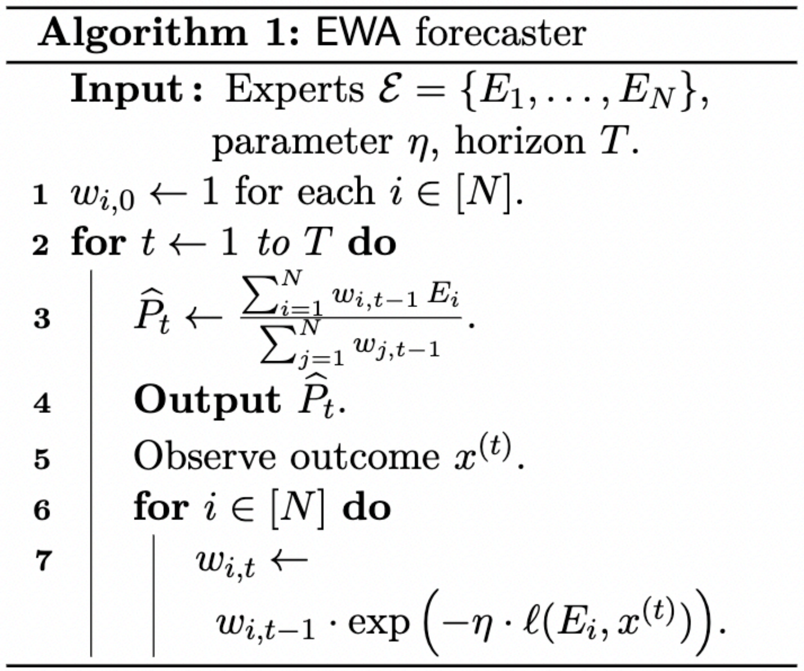
Run an EWA / RWM-based online learning algorithm with $\mathscr{N}$ as the set of experts.

- EWA = Exponential Weighted Average (returns mixture of $\mathscr{N}$-distributions).
- RWM = Randomized Weighted Majority (returns single distribution in $\mathscr{N}$).

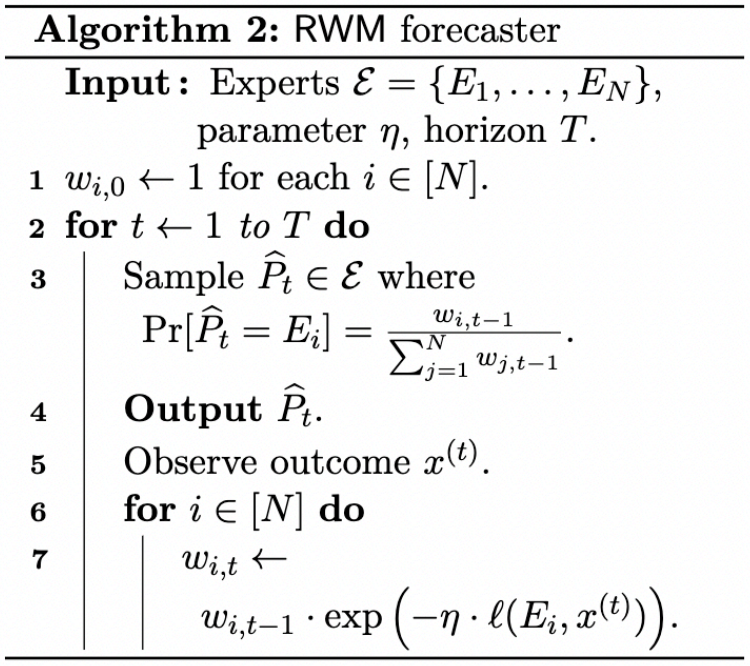
Use regret bounds of EWA/RWM to get learning guarantees.

EWA and RWM based Learning Algorithms

$$w_{i,t} = \exp\left(-\eta \sum_{s=1}^{t} \ell(E_i, x^{(s)})\right)$$



```
Algorithm 1: EWA forecaster
Input: Experts $\mathcal{E} = \{E_1, \dots, E_N\}$,
       parameter $\eta$, horizon $T$.
1 $w_{i,0} \leftarrow 1$ for each $i \in [N]$.
2 for $t \leftarrow 1$ to $T$ do
3     $\widehat{P}_t \leftarrow \frac{\sum_{i=1}^{N} w_{i,t-1} E_i}{\sum_{j=1}^{N} w_{j,t-1}}$.
4     Output $\widehat{P}_t$.
5     Observe outcome $x^{(t)}$.
6     for $i \in [N]$ do
7         $w_{i,t} \leftarrow$
            $w_{i,t-1} \cdot \exp\left(-\eta \cdot \ell(E_i, x^{(t)})\right)$.
```



```
Algorithm 2: RWM forecaster
Input: Experts $\mathcal{E} = \{E_1, \dots, E_N\}$,
       parameter $\eta$, horizon $T$.
1 $w_{i,0} \leftarrow 1$ for each $i \in [N]$.
2 for $t \leftarrow 1$ to $T$ do
3     Sample $\widehat{P}_t \in \mathcal{E}$ where
        $\Pr[\widehat{P}_t = E_i] = \frac{w_{i,t-1}}{\sum_{j=1}^{N} w_{j,t-1}}$.
4     Output $\widehat{P}_t$.
5     Observe outcome $x^{(t)}$.
6     for $i \in [N]$ do
7         $w_{i,t} \leftarrow$
            $w_{i,t-1} \cdot \exp\left(-\eta \cdot \ell(E_i, x^{(t)})\right)$.
```

Regret Bounds of EWA & RWM

$$Reg_T(\mathscr{A};\mathscr{E}) = \sum_{t=1}^{T} \ell(\hat{P}_t, x^{(t)}) - \min_{E\in\mathscr{E}} \sum_{t=1}^{T} \ell(P, x^{(t)})$$

$$\ell(P,x) = \log\left(\frac{1}{P(x)}\right)$$

For finite $\mathscr{E}$, EWA forecaster gives

$$Reg_T(\text{EWA};\mathscr{E}) \le O(\log N)$$

For finite $\mathscr{E}$, RWM algorithm gives

$$Reg_T(\text{RWM};\mathscr{E})] \le O(\sqrt{T\log N})$$

Our Algo Framework: Time Efficiency

Goal is to implement the EWA/RWM based algorithms efficiently.

- EWA/RWM based algorithms use exponential space and time even when k and d are constant.
- Number of candidate distribution is $O\left(\left(\frac{nk}{\varepsilon}\right)^{nk^{d+1}}\right)$.
- For trees, chordal graphs etc., one can take advantage of the product structure of the EWA/RWM mixture distribution.
- Can sample efficiently from it “edge-by-edge” (each element of $\mathcal{N}$ is a Bayes net $(G, \mathbb{P})$).

Efficient Learning of Tree-structured distributions

Consider $\mathcal{N}^{\text{TREE}}$, the discretization of all tree-structured Bayes nets $(T, \mathbb{P} = \{p_1, \ldots, p_n\})$.

- Each spanning tree T of K_n is oriented (outwards) with root 1.

Algorithm

1. Sample p_1 (distribution at root node) from discretization.
2. Sample tree structure T.
3. Sample $p_i(x_i \mid x_{pa(i)})$ from discreatization for every $i \in \{2, \ldots, n\}$.

- Steps (1) and (3) involve sampling from Poly$(n, \frac{1}{\varepsilon})$ possibilities for constant k.

Tree-structured distributions Contd.

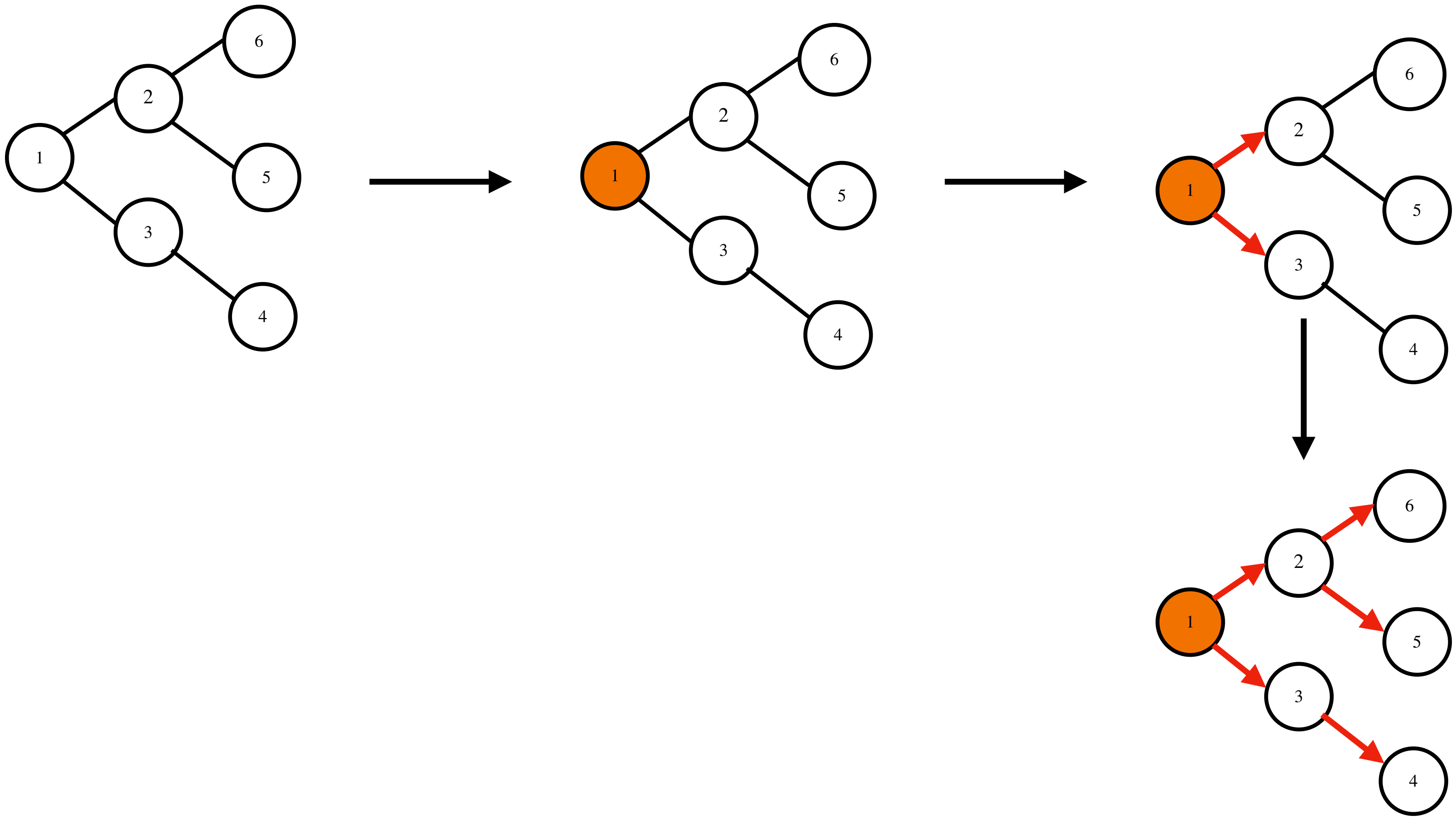
- The sampling step involves computing a normalization factor with exponential terms of the form

$$\sum_{\substack{T \\ \text{out oriented}}} \prod_{e \in T} \mathrm{wt}(e)$$

- Can efficiently compute this using a weighted version of Tutte's matrix tree theorem as the determinant of associated Laplacian matrix (DL 20).

- The probability of sampling an edge of T is the ratio of two Laplacian determinants.

Sampling Tree Structured Distribution Example



Learning Chordal Distributions

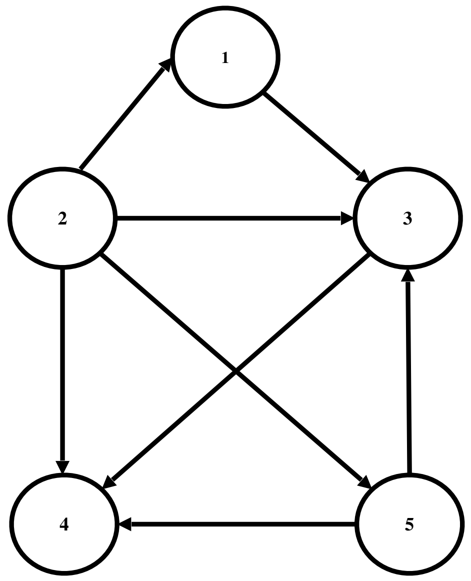
- Unlike trees, not all orientations of a chordal skeleton G are acyclic.

- Need to compute weighted sums over all partial *acyclic orientations* of G consistent with a particular sub-orientation.

  - This generalizes the problem of "counting acyclic orientations of a chordal graph" **(BS 22)**.

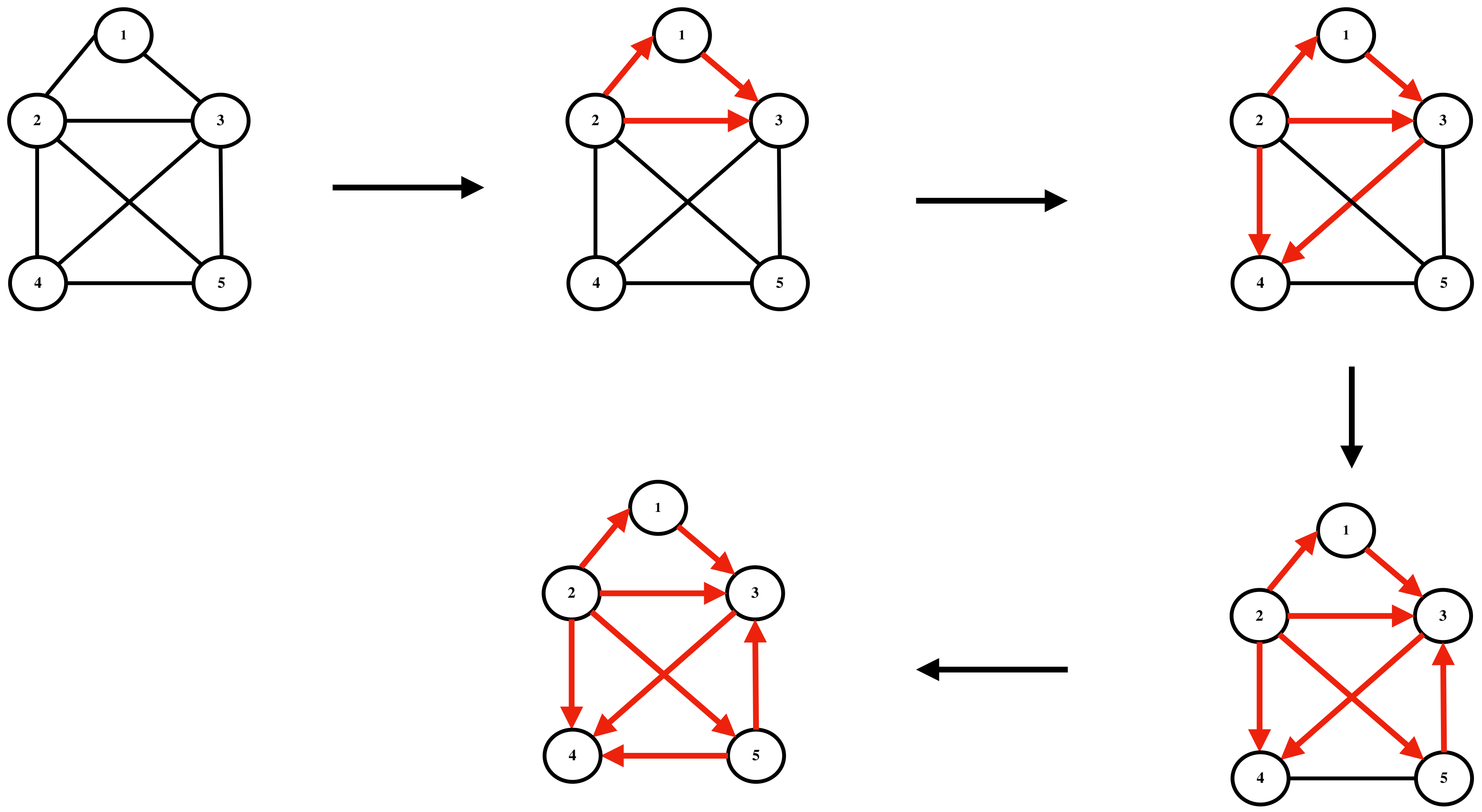
  - The **clique tree decomposition** of a chordal graph guides the DP computation and sampling.

- DP table can be used to sample a random acyclic orientation, giving a random DAG $\overrightarrow{G}$ with appropriate probability.



Chordal-structured DAG

Sampling Chordal distribution Example



Conclusion

- Designed the first efficient algorithm for learning chordal-structured distribution.
- Our approach gives new algorithm for learning tree structured distributions.
- Can our bounds be improved?
- Can this approach be extended for other models?

Thank You!