Learning Tractable Graphical Models: Latent Trees and Tree Mixtures

Anima Anandkumar

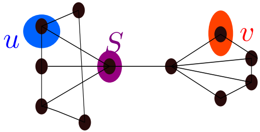
U.C. Irvine

Joint work with Furong Huang, U.N. Niranjan, Daniel Hsu and Sham Kakade.

High-Dimensional Graphical Modeling

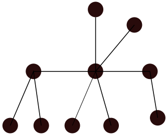
Modeling Conditional Independencies through Graphs

- $X_u \perp\!\!\!\perp X_v | X_S$.
- Learning and inference are NP-hard.



Tractable Models: Tree Models

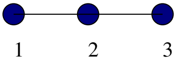
- Efficient inference using belief propagation



Walk-up: Learning Tree Models

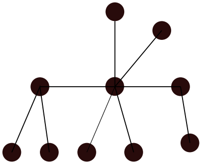
Data processing inequality for Markov chains

$$I(X_1; X_3) \leq I(X_1; X_2), I(X_2; X_3).$$



Tree Structure Estimation (Chow and Liu '68)

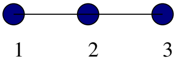
- MLE: Max-weight tree with estimated mutual information weights



Walk-up: Learning Tree Models

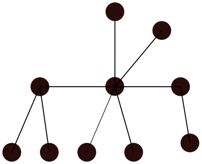
Data processing inequality for Markov chains

$$I(X_1; X_3) \leq I(X_1; X_2), I(X_2; X_3).$$



Tree Structure Estimation (Chow and Liu '68)

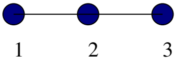
- MLE: Max-weight tree with estimated mutual information weights
- Pairwise statistics suffice



Walk-up: Learning Tree Models

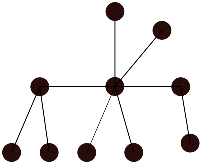
Data processing inequality for Markov chains

$$I(X_1; X_3) \leq I(X_1; X_2), I(X_2; X_3).$$



Tree Structure Estimation (Chow and Liu '68)

- MLE: Max-weight tree with estimated mutual information weights
- Pairwise statistics suffice
- n samples and p nodes

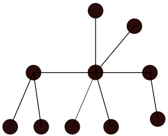


Sample complexity: $\frac{\log p}{n} = O(1)$.

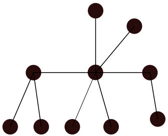
Learning Tractable Graphical Models

Tractable Models: Tree Models

- Efficient inference using belief propagation
- MLE is easy to compute.
- Tree models are highly restrictive.

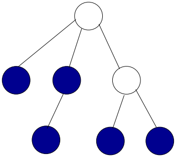


Learning Tractable Graphical Models



Tractable Models: Tree Models

- Efficient inference using belief propagation
- MLE is easy to compute.
- Tree models are highly restrictive.

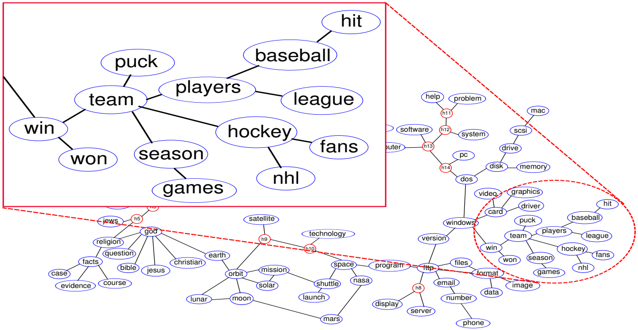


Latent tree graphical models

- Tree models with hidden variables.
- Number and location of hidden variables unknown.

Application: Hierarchical Topic Modeling

- Data: Word co-occurrences.
- Graph: Topic-word structure.



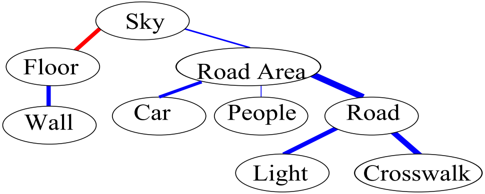
Application of Latent Trees: Object Recognition

- Challenge: Succinct representation of large-scale data
  - Input: ~ 100 object categories, ~ 4000 training images
  - Goal: learn $\sim 2^{100}$ co-occurrence probabilities
- Solution: Latent tree graphical models

"Context Models and Out-of-context Objects," M. J. Choi, A. Torralba, and A. S. Willsky, Pattern Recognition Letters, 2012.

Application of Latent Trees: Object Recognition

- Challenge: Succinct representation of large-scale data
  - Input: ~ 100 object categories, ~ 4000 training images
  - Goal: learn $\sim 2^{100}$ co-occurrence probabilities
- Solution: Latent tree graphical models

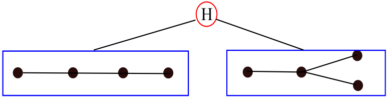


"Context Models and Out-of-context Objects," M. J. Choi, A. Torralba, and A. S. Willsky, Pattern Recognition Letters, 2012.

Tree Mixture Models

Tree Mixture Models

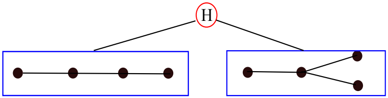
- Multiple graphs: context specific dependencies
- Each component is a tree model
- Unsupervised learning: Class variable is latent or hidden



Tree Mixture Models

Tree Mixture Models

- Multiple graphs: context specific dependencies
- Each component is a tree model
- Unsupervised learning: Class variable is latent or hidden



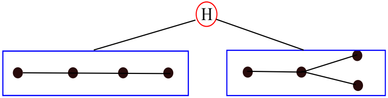
Why use tree mixtures?

- Efficient Inference: BP on component trees and combining them.
- Similarly marginalization and sampling also efficient.

Tree Mixture Models

Tree Mixture Models

- Multiple graphs: context specific dependencies
- Each component is a tree model
- Unsupervised learning: Class variable is latent or hidden



Why use tree mixtures?

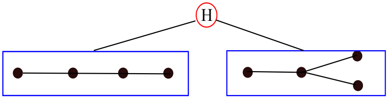
- Efficient Inference: BP on component trees and combining them.
- Similarly marginalization and sampling also efficient.

Learning: Alternatives to EM (Meila & Jordan)?

Tree Mixture Models

Tree Mixture Models

- Multiple graphs: context specific dependencies
- Each component is a tree model
- Unsupervised learning: Class variable is latent or hidden



Why use tree mixtures?

- Efficient Inference: BP on component trees and combining them.
- Similarly marginalization and sampling also efficient.

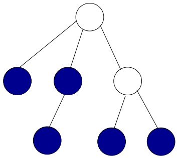
Learning: Alternatives to EM (Meila & Jordan)?

In this talk: learning latent tree models and tree mixtures.

Summary of Results

Latent Tree Models

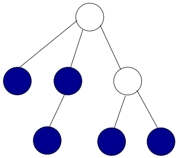
- Number of hidden variables and location unknown
- Integrated structure and parameter estimation.
- Local learning with global consistency guarantees.



Summary of Results

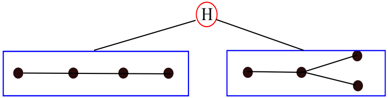
Latent Tree Models

- Number of hidden variables and location unknown
- Integrated structure and parameter estimation.
- Local learning with global consistency guarantees.



Mixtures of Trees

- Structure and parameters under different contexts unknown
- Unsupervised setting: choice variable hidden.
- Efficient methods for consistent structure and parameter learning.



Previous Approaches

Algorithms for Structure Estimation

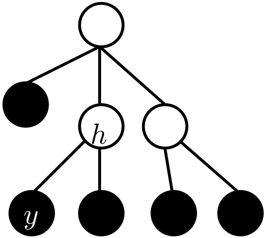
- Chow and Liu (68): Tree estimation
- Meinshausen and Bühlmann (06): Convex relaxation
- Ravikumar, Wainwright, Lafferty (10): Convex relaxation
- Bresler, Mossel and Sly (09): Bounded-degree graphs ...

Learning with Hidden Variables

- Erdös, et. al. (99): Latent trees
- Daskalakis, Mossel and Roch (06): Latent trees
- Choi, Tan, Anandkumar and Willsky (10): Latent trees
- Chandrasekaran, Parrilo and Willsky (11): Latent Gaussian models,
- Anandkumar et. al (12): Tensor decompositions ...

Outline

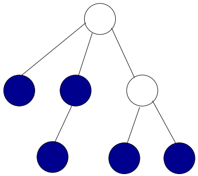
Learning Latent Tree Graphical Models



Linear Multivariate Models

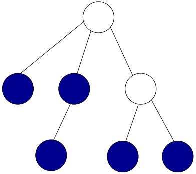
- Conditional independence w.r.t tree
- Categorical k-state hidden variables.
- Multivariate d-dimensional observed variables. $k \leq d$.
- When y is nbr. of h, $\mathbb{E}[y|h] = Ah$.
- Includes discrete, Poisson and Gaussian models, Gaussian mixtures etc.

Additive Tree Distances



Information Distances $[d_{i,j}]$ for Tree Models

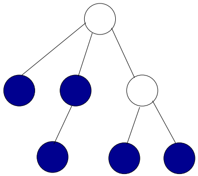
Additive Tree Distances



Information Distances $[d_{i,j}]$ for Tree Models

Gaussian scalar: $d_{ij} := -\log|\rho_{ij}|$.

Additive Tree Distances

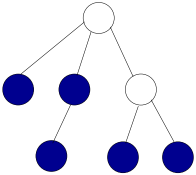


Information Distances $[d_{i,j}]$ for Tree Models

Gaussian scalar: $d_{ij} := -\log|\rho_{ij}|$.

Linear multivariate models: $d_{ij} := -\log \dfrac{\prod_{\sigma_i \neq 0} \sigma(\mathbb{E}[y_i y_j^\top])}{\sqrt{\det \mathbb{E}[y_i y_i^\top] \det \mathbb{E}[y_j y_j^\top]}}$.

Additive Tree Distances



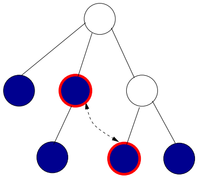
Information Distances $[d_{i,j}]$ for Tree Models

Gaussian scalar: $d_{ij} := -\log|\rho_{ij}|$.

Linear multivariate models: $d_{ij} := -\log \frac{\prod_{\sigma_i \neq 0} \sigma(\mathbb{E}[y_i y_j^\top])}{\sqrt{\det \mathbb{E}[y_i y_i^\top] \det \mathbb{E}[y_j y_j^\top]}}$.

$[d_{i,j}]$ is an additive tree metric: $\boxed{d_{k,l} = \sum_{(i,j) \in \mathrm{Path}(k,l;E)} d_{i,j}.}$

Additive Tree Distances



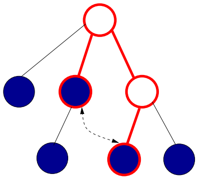
Information Distances $[d_{i,j}]$ for Tree Models

Gaussian scalar: $d_{ij} := -\log|\rho_{ij}|$.

Linear multivariate models: $d_{ij} := -\log \frac{\prod_{\sigma_i \neq 0} \sigma(\mathbb{E}[y_i y_j^\top])}{\sqrt{\det \mathbb{E}[y_i y_i^\top] \det \mathbb{E}[y_j y_j^\top]}}$.

$[d_{i,j}]$ is an additive tree metric: $\boxed{d_{k,l} = \sum_{(i,j)\in \text{Path}(k,l;E)} d_{i,j}.}$

Additive Tree Distances



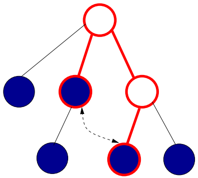
Information Distances $[d_{i,j}]$ for Tree Models

Gaussian scalar: $d_{ij} := -\log|\rho_{ij}|$.

Linear multivariate models: $d_{ij} := -\log \dfrac{\prod_{\sigma_i \neq 0} \sigma(\mathbb{E}[y_i y_j^\top])}{\sqrt{\det \mathbb{E}[y_i y_i^\top] \det \mathbb{E}[y_j y_j^\top]}}$.

$[d_{i,j}]$ is an additive tree metric: $\boxed{d_{k,l} = \sum_{(i,j) \in \mathrm{Path}(k,l;E)} d_{i,j}.}$

Additive Tree Distances



Information Distances $[d_{i,j}]$ for Tree Models

Gaussian scalar: $d_{ij} := -\log|\rho_{ij}|$.

Linear multivariate models: $d_{ij} := -\log \dfrac{\prod_{\sigma_i \neq 0} \sigma(\mathbb{E}[y_i y_j^\top])}{\sqrt{\det \mathbb{E}[y_i y_i^\top] \det \mathbb{E}[y_j y_j^\top]}}$.

$[d_{i,j}]$ is an additive tree metric: $\boxed{d_{k,l} = \sum_{(i,j)\in \mathrm{Path}(k,l;E)} d_{i,j}.}$

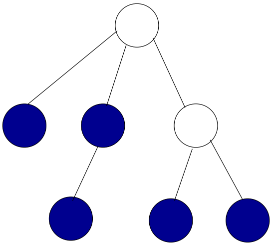
Learning latent tree using $[\hat{d}_{i,j}]$

Siblings Test Based on Information Distances

Exact Statistics: Distances $[d_{i,j}]$

Let $\Phi_{ijk} := d_{i,k} - d_{j,k}$.

- $-d_{i,j} < \Phi_{ijk} = \Phi_{ijk'} < d_{i,j}\ \forall\, k, k' \neq i, j,$ $\iff$ i, j leaves with common parent
- $\Phi_{ijk} = d_{i,j},\ \forall\, k \neq i, j,$ $\iff$ i is a leaf and j is its parent.



Sample Statistics: ML Estimates $[\hat{d}_{i,j}]$

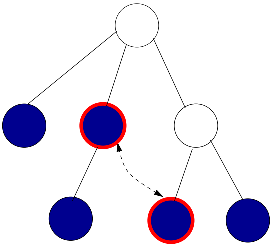
Use only short distances: $d_{i,k}, d_{j,k} < \tau$, Relax equality relationships

Siblings Test Based on Information Distances

Exact Statistics: Distances $[d_{i,j}]$

Let $\Phi_{ijk} := d_{i,k} - d_{j,k}$.

- $-d_{i,j} < \Phi_{ijk} = \Phi_{ijk'} < d_{i,j} \; \forall \, k, k' \neq i, j,$ $\iff$ i, j leaves with common parent
- $\Phi_{ijk} = d_{i,j}, \; \forall \, k \neq i, j,$ $\iff$ i is a leaf and j is its parent.



Sample Statistics: ML Estimates $[\hat{d}_{i,j}]$

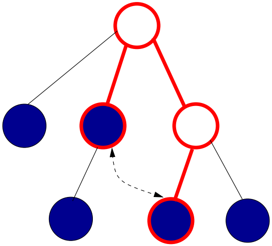
Use only short distances: $d_{i,k}, d_{j,k} < \tau$, Relax equality relationships

Siblings Test Based on Information Distances

Exact Statistics: Distances $[d_{i,j}]$

Let $\Phi_{ijk} := d_{i,k} - d_{j,k}$.

- $-d_{i,j} < \Phi_{ijk} = \Phi_{ijk'} < d_{i,j} \; \forall\, k, k' \neq i, j, \iff i, j$ leaves with common parent
- $\Phi_{ijk} = d_{i,j}, \; \forall\, k \neq i, j, \iff i$ is a leaf and j is its parent.



Sample Statistics: ML Estimates $[\hat{d}_{i,j}]$

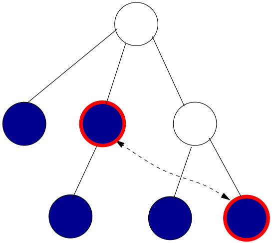
Use only short distances: $d_{i,k}, d_{j,k} < \tau$, Relax equality relationships

Siblings Test Based on Information Distances

Exact Statistics: Distances $[d_{i,j}]$

Let $\Phi_{ijk} := d_{i,k} - d_{j,k}$.

- $-d_{i,j} < \Phi_{ijk} = \Phi_{ijk'} < d_{i,j}\ \forall\, k, k' \neq i, j,$ $\iff$ i, j leaves with common parent
- $\Phi_{ijk} = d_{i,j},\ \forall\, k \neq i, j,$ $\iff$ i is a leaf and j is its parent.



Sample Statistics: ML Estimates $[\hat{d}_{i,j}]$

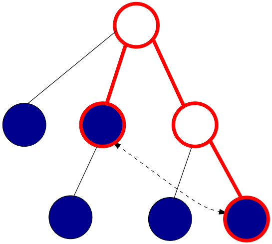
Use only short distances: $d_{i,k}, d_{j,k} < \tau$, Relax equality relationships

Siblings Test Based on Information Distances

Exact Statistics: Distances $[d_{i,j}]$

Let $\Phi_{ijk} := d_{i,k} - d_{j,k}$.

- $-d_{i,j} < \Phi_{ijk} = \Phi_{ijk'} < d_{i,j} \; \forall\, k, k' \neq i, j,$ $\iff$ i, j leaves with common parent
- $\Phi_{ijk} = d_{i,j}, \; \forall\, k \neq i, j,$ $\iff$ i is a leaf and j is its parent.



Sample Statistics: ML Estimates $[\hat{d}_{i,j}]$

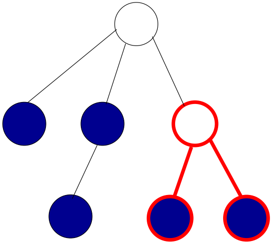
Use only short distances: $d_{i,k}, d_{j,k} < \tau$, Relax equality relationships

Siblings Test Based on Information Distances

Exact Statistics: Distances $[d_{i,j}]$

Let $\Phi_{ijk} := d_{i,k} - d_{j,k}$.

- $-d_{i,j} < \Phi_{ijk} = \Phi_{ijk'} < d_{i,j}\ \forall\, k, k' \neq i, j,\ \iff\ i, j$ leaves with common parent
- $\Phi_{ijk} = d_{i,j},\ \forall\, k \neq i, j,\ \iff\ i$ is a leaf and j is its parent.



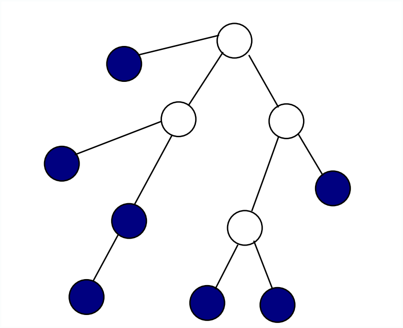
Sample Statistics: ML Estimates $[\hat{d}_{i,j}]$

Use only short distances: $d_{i,k}, d_{j,k} < \tau$, Relax equality relationships

Recursive Grouping

Recursive Grouping Algorithm (Choi, Tan, **A.**, Willsky)

- Sibling test and remove leaves
- Build tree from bottom up

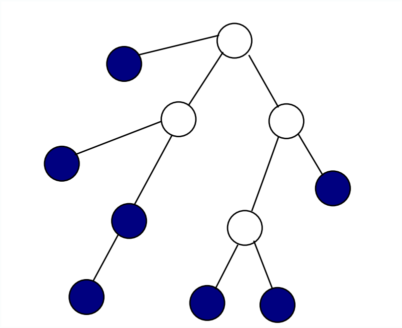


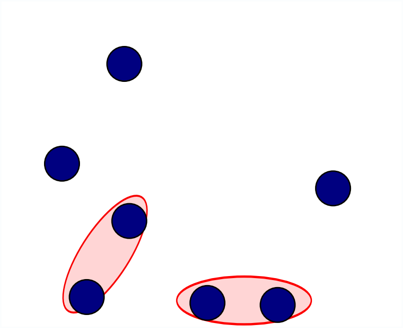
- Consistent structure estimation.
- Serial method, high computational complexity.

Recursive Grouping

Recursive Grouping Algorithm (Choi, Tan, **A.**, Willsky)

- Sibling test and remove leaves
- Build tree from bottom up



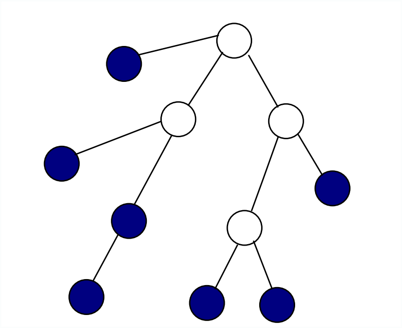


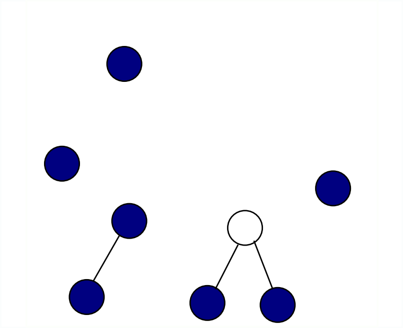
- Consistent structure estimation.
- Serial method, high computational complexity.

Recursive Grouping

Recursive Grouping Algorithm (Choi, Tan, **A.**, Willsky)

- Sibling test and remove leaves
- Build tree from bottom up



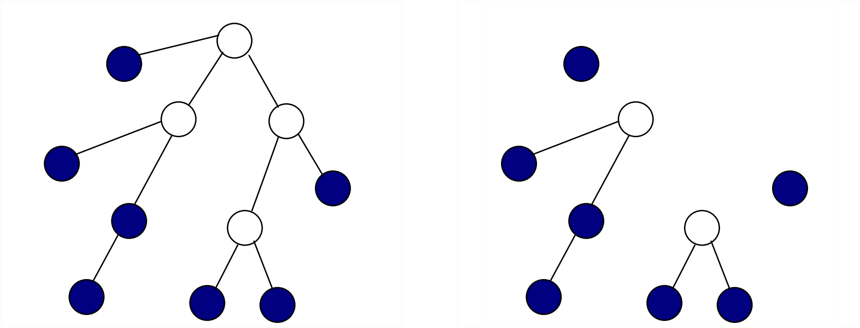


- Consistent structure estimation.
- Serial method, high computational complexity.

Recursive Grouping

Recursive Grouping Algorithm (Choi, Tan, **A.**, Willsky)

- Sibling test and remove leaves
- Build tree from bottom up

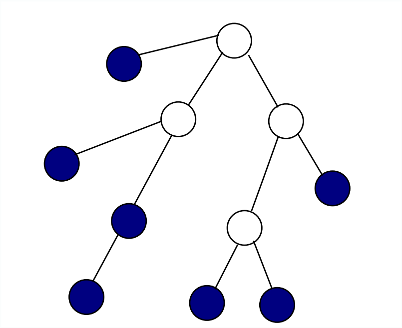


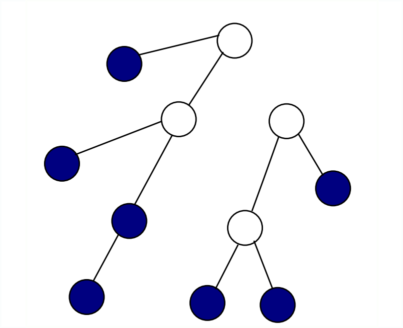
- Consistent structure estimation.
- Serial method, high computational complexity.

Recursive Grouping

Recursive Grouping Algorithm (Choi, Tan, **A.**, Willsky)

- Sibling test and remove leaves
- Build tree from bottom up



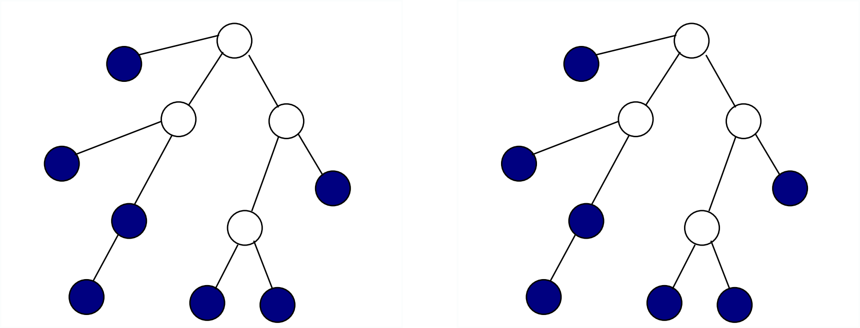


- Consistent structure estimation.
- Serial method, high computational complexity.

Recursive Grouping

Recursive Grouping Algorithm (Choi, Tan, **A.**, Willsky)

- Sibling test and remove leaves
- Build tree from bottom up



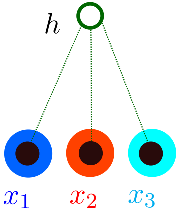
- Consistent structure estimation.
- Serial method, high computational complexity.

Outline

Overview of Proposed Parameter Learning Method

Toy Model: 3-star

- Linear multivariate model.
- $A^r_{x_i|h} := \mathbb{E}(x_i|h = e_r)$. and $\lambda_r := \mathbb{P}[h = e_r]$.



$$\boxed{\mathbb{E}(x_1 \otimes x_2 \otimes x_3) = \sum_{r=1}^{k} \lambda_r A^r_{x_1|h} \otimes A^r_{x_2|h} \otimes A^r_{x_3|h}.}$$

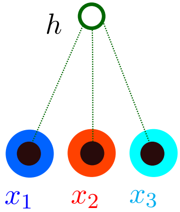
Guaranteed Recovery through Tensor Decomposition

- Transition matrices $A_{x_i|h}$ have full column rank.
- Linear algebraic operations: SVD and tensor power iterations.

"Tensor Decompositions for Learning Latent Variable Models" by A. Anandkumar, R. Ge, D. Hsu, S.M. Kakade and M. Telgarsky. Preprint, October 2012.

Overview of Tensor Decomposition Technique

- Let $a_r = \mathbb{E}(x_i | h = e_r)$ for all i and $\lambda_r := \mathbb{P}[h = e_r]$.
- $M_3 = \mathbb{E}[x_1 \otimes x_2 \otimes x_3] = \sum_{i=1}^{k} \lambda_i a_i^{\otimes 3}$.



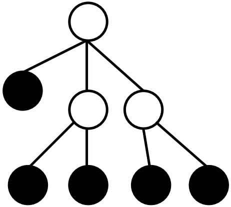
Intuition: if a_i are orthogonal

- $M_3(I, a_1, a_1) := \sum_i \lambda_i \langle a_i, a_1 \rangle^2 a_i = \lambda_1 a_1$.
- a_i are eigenvectors of the tensor M_3.

Convert to an orthogonal tensor using pairwise moments

- $M_2 := \mathbb{E}[x_1 \otimes x_2] = \sum_i \lambda_i a_i^{\otimes 2}$.
- Whitening matrix: $W^\top M_2 W = I$.
- Consider tensor $M_3(W, W, W) := \sum_i \lambda_i (W^\top a_i)^{\otimes 3}$. It is an orthogonal tensor.

Parameter Learning in Latent Trees



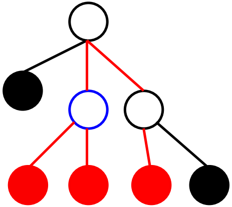
Learning through Hierarchical Tensor Decomposition

- Assume known tree structure.
- Decompose different triplets: hidden variable is join point on tree.

Alignment issue

- Tensor decomposition is an unsupervised method.
- Hidden labels permuted across different triplets.
- Solution: Align using common node in triplets.

Parameter Learning in Latent Trees



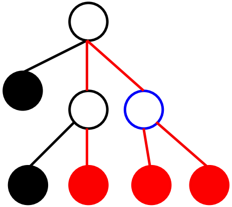
Learning through Hierarchical Tensor Decomposition

- Assume known tree structure.
- Decompose different triplets: hidden variable is join point on tree.

Alignment issue

- Tensor decomposition is an unsupervised method.
- Hidden labels permuted across different triplets.
- Solution: Align using common node in triplets.

Parameter Learning in Latent Trees



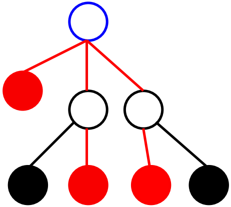
Learning through Hierarchical Tensor Decomposition

- Assume known tree structure.
- Decompose different triplets: hidden variable is join point on tree.

Alignment issue

- Tensor decomposition is an unsupervised method.
- Hidden labels permuted across different triplets.
- Solution: Align using common node in triplets.

Parameter Learning in Latent Trees



Learning through Hierarchical Tensor Decomposition

- Assume known tree structure.
- Decompose different triplets: hidden variable is join point on tree.

Alignment issue

- Tensor decomposition is an unsupervised method.
- Hidden labels permuted across different triplets.
- Solution: Align using common node in triplets.

Outline

Integrated Learning

So far..

- Consistent structure learning through sibling tests on distances.
- Parameter learning through tensor decomposition on triplets.

Challenges

- How to integrate structure and parameter learning?
- Can we save on computations through integration?
- Can we learn parameters as we learn the structure?
- Can we parallelize learning for scalability?

Integrated Learning

So far..

- Consistent structure learning through sibling tests on distances.
- Parameter learning through tensor decomposition on triplets.

Challenges

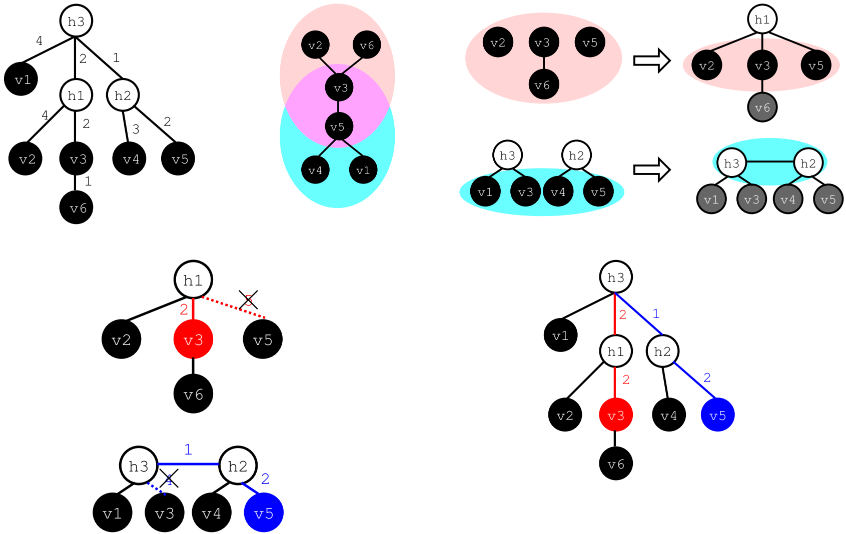
- How to integrate structure and parameter learning?
- Can we save on computations through integration?
- Can we learn parameters as we learn the structure?
- Can we parallelize learning for scalability?

Key Ideas

- Divide and conquer: find (overlapping) groups of observed variables.
- Learn local subtrees (and parameters) over the groups independently.
- Merge subtrees and tweak parameters to obtain global latent tree model.

Parallel Chow-Liu Based Grouping Algorithm

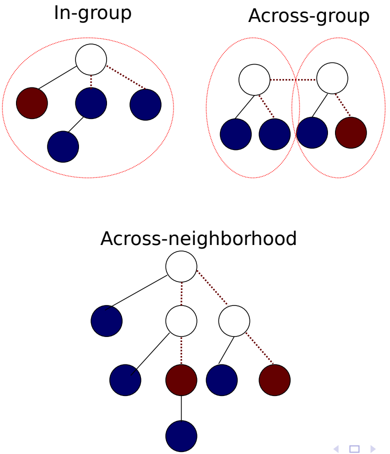
Minimum spanning tree using information distance $[\hat{d}_{i,j}]$.



Alignment of Parameters

Alignment Correction

- In-group
- Across-group
- Across-neighborhood



Consistency Guarantees

Theorem

The proposed method consistently recovers the structure with $O(\log p)$ samples and parameters with $\text{poly}(p)$ samples.

Extent of parallelism

- Size of groups $\boxed{\Gamma \leq \Delta^{1+\frac{u}{l}\delta}.}$
- Effective depth $\delta := \max_i\{\min_j\{\mathsf{path}(v_i, v_j; \mathcal{T})\}$.
- Maximum degree in latent tree: Δ.
- Upper and lower bound on distances between neighbors in the latent tree: u and l.

Implications

- For homogeneous HMM, constant sized groups.
- Worst case: star graphs.

Computational Complexity

- N samples, d dimensional observed variables, k state hidden variables.
- p number of observed variables. z non-zero entries per sample.
- Γ sized groups.

Algorithm Steps	Time/worker	Degree of parallelism
Information Distance Estimation	$O(Nz + d + k^3)$	$O(p^2)$
Structure: Minimum Spanning Tree	$O(\log p)$	$O(p^2)$
Structure: Local Recursive Grouping	$O(\Gamma^3)$	$O(p/\Gamma)$
Parameter: Tensor Decomposition	$O(\Gamma k^3 + \Gamma d k^2)$	$O(p/\Gamma)$
Merging and Alignment Correction	$O(dk^2)$	$O(p/\Gamma)$

"Integrated Structure and Parameter Learning in Latent Tree Graphical Models" by F. Huang, U. N. Niranjan, A. Anandkumar. Preprint, June 2014.

Proof Ideas

Relating Chow-Liu Tree with Latent Tree

- Surrogate $\mathrm{Sg}(i)$ for node i: observed node with strongest correlation

$$\mathrm{Sg}(i) := \operatorname*{argmin}_{j \in V} d_{i,j}$$

- Neighborhood preservation

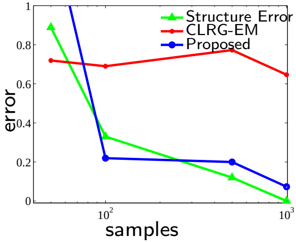
$$(i,j) \in T \Rightarrow (\mathrm{Sg}(i), \mathrm{Sg}(j)) \in T_{\mathrm{ML}}.$$

Chow-Liu grouping reverses edge contractions

Proof by induction

Experiments

- $d = k = 2$ dimensions, $p = 9$ number of variables.



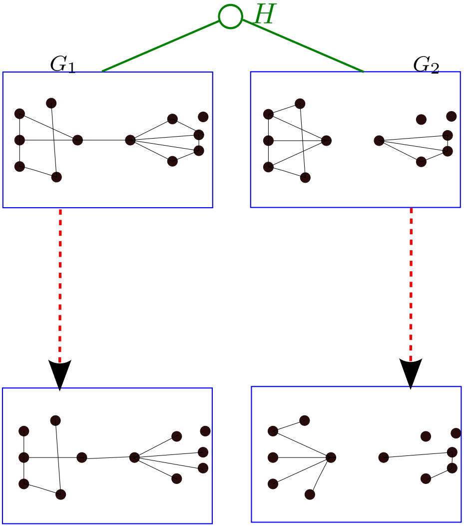
d	p	N	Struct Error	Param Error	Running Time(s)
10	9	50K	0	0.0104	3.8
100	9	50K	0	0.0967	4.4
1000	9	50K	0	0.1014	5.1
10,000	9	50K	0	0.0917	29.9
100,000	9	50k	0	0.0812	56.5
100	9	50K	0	0.0967	10.9
100	81	50K	0.06	0.1814	323.7
100	729	50K	0.16	0.1913	4220.1

Outline

Mixtures of Graphical Models: Our Approach

Our Approach

- Consider data from graphical model mixture
- Output tree mixture: best tree approx. of each component

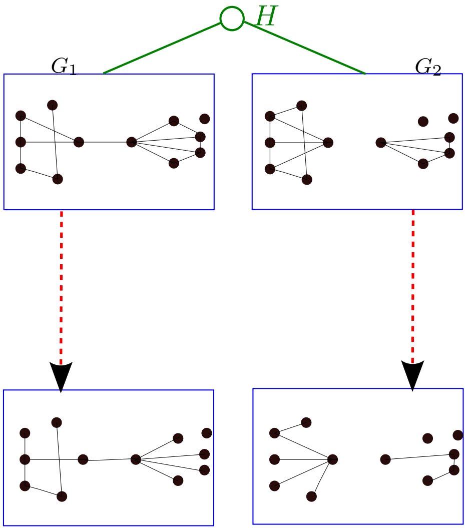


Mixtures of Graphical Models: Our Approach

Our Approach

- Consider data from graphical model mixture
- Output tree mixture: best tree approx. of each component

Steps



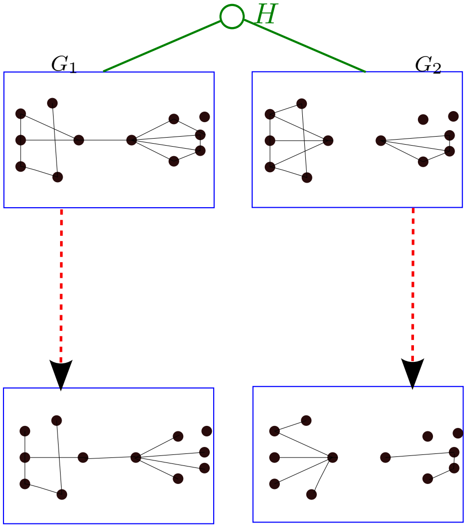
Mixtures of Graphical Models: Our Approach

Our Approach

- Consider data from graphical model mixture
- Output tree mixture: best tree approx. of each component

Steps

- Estimation of union graph



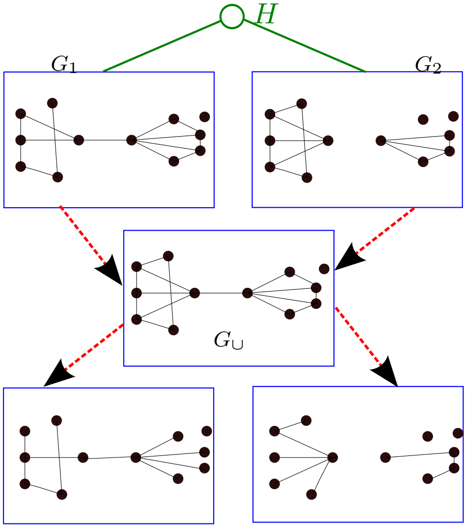
Mixtures of Graphical Models: Our Approach

Our Approach

- Consider data from graphical model mixture
- Output tree mixture: best tree approx. of each component

Steps

- Estimation of union graph



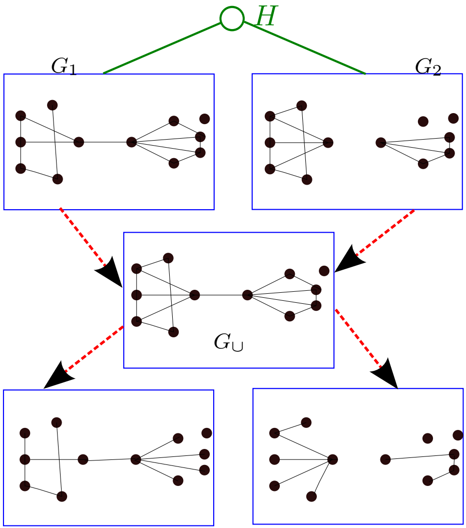
Mixtures of Graphical Models: Our Approach

Our Approach

- Consider data from graphical model mixture
- Output tree mixture: best tree approx. of each component

Steps

- Estimation of union graph
- Estimation of pairwise moments in each component



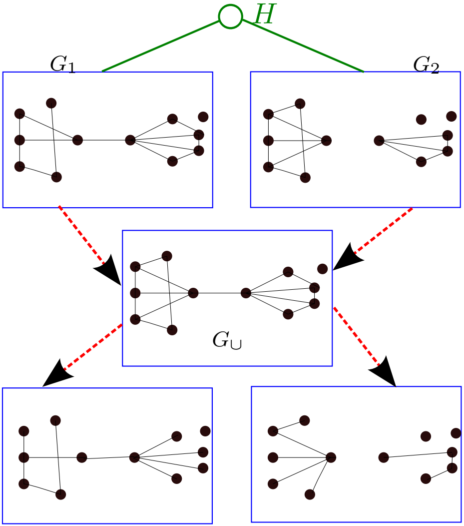
Mixtures of Graphical Models: Our Approach

Our Approach

- Consider data from graphical model mixture
- Output tree mixture: best tree approx. of each component

Steps

- Estimation of union graph
- Estimation of pairwise moments in each component
- Tree approximation of each component via Chow-Liu algorithm.



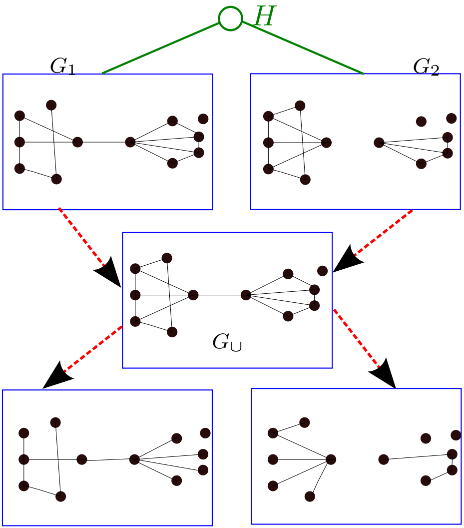
Mixtures of Graphical Models: Our Approach

Our Approach

- Consider data from graphical model mixture
- Output tree mixture: best tree approx. of each component

Steps

- Estimation of union graph
- Estimation of pairwise moments in each component
- Tree approximation of each component via Chow-Liu algorithm.

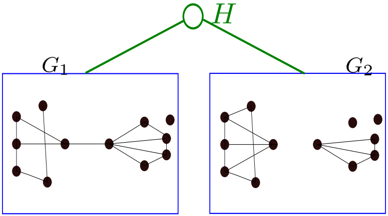


Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

Challenges

- Cannot reduce to triplet tensor decomposition



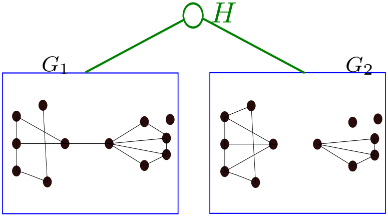
Solutions

Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

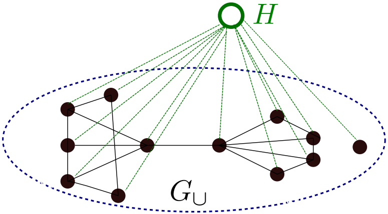
Challenges

- Cannot reduce to triplet tensor decomposition



Solutions

- $G_\cup$: union graph learnt from rank test

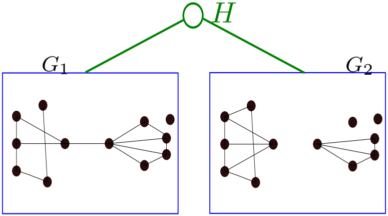


Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

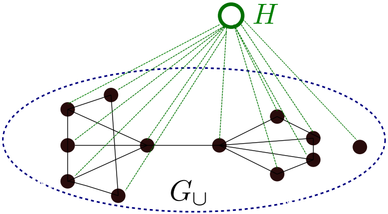
Challenges

- Cannot reduce to triplet tensor decomposition



Solutions

- $G_\cup$: union graph learnt from rank test
- Use separators on union graph $G_\cup$:

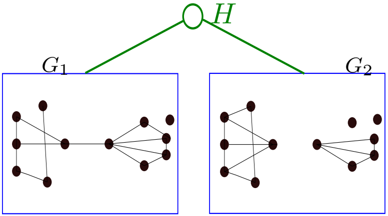


Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

Challenges

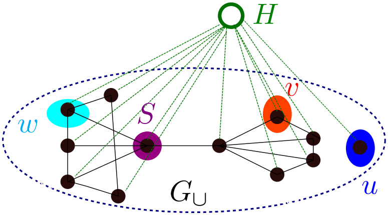
- Cannot reduce to triplet tensor decomposition



Solutions

- $G_\cup$: union graph learnt from rank test
- Use separators on union graph $G_\cup$:

$$\boxed{X_u \perp\!\!\!\perp X_v \perp\!\!\!\perp X_w \,|X_S, H}$$

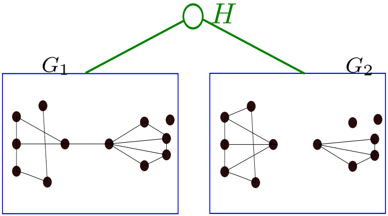


Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

Challenges

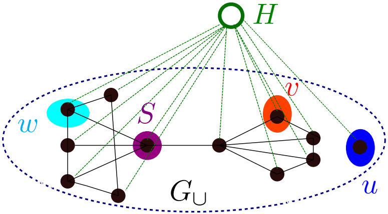
- Cannot reduce to triplet tensor decomposition
- Need to learn higher order moments of mixture components.



Solutions

- $G_\cup$: union graph learnt from rank test
- Use separators on union graph $G_\cup$:

$$X_u \perp\!\!\!\perp X_v \perp\!\!\!\perp X_w \,|X_S, H$$

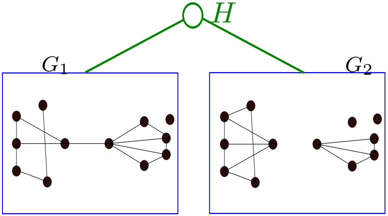


Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

Challenges

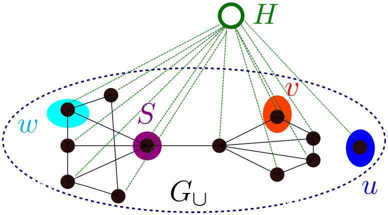
- Cannot reduce to triplet tensor decomposition
- Need to learn higher order moments of mixture components.



Solutions

- $G_\cup$: union graph learnt from rank test
- Use separators on union graph $G_\cup$:

  $X_u \perp\!\!\!\perp X_v \perp\!\!\!\perp | X_S, H$
- Learn tree mixture approximation: estimate pairwise mixture moments

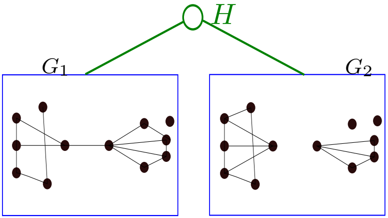


Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

Challenges

- Cannot reduce to triplet tensor decomposition
- Need to learn higher order moments of mixture components.

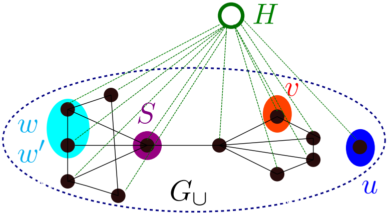


Solutions

- $G_\cup$: union graph learnt from rank test
- Use separators on union graph $G_\cup$:

$$\boxed{X_u \perp\!\!\!\perp X_v \perp\!\!\!\perp X_{w,w'} | X_S, H}$$

- Learn tree mixture approximation: estimate pairwise mixture moments

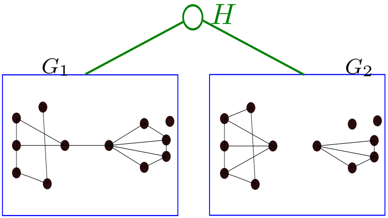


Learning Graphical Model Mixtures

Adapt Tensor Decomposition Method for Graphical Model Mixtures?

Challenges

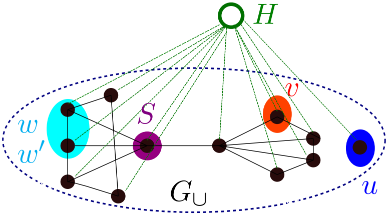
- Cannot reduce to triplet tensor decomposition
- Need to learn higher order moments of mixture components.



Solutions

- $G_\cup$: union graph learnt from rank test
- Use separators on union graph $G_\cup$:

  $X_u \perp\!\!\!\perp X_v \perp\!\!\!\perp X_{w,w'} | X_S, H$
- Learn tree mixture approximation: estimate pairwise mixture moments



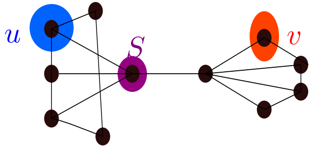
Efficient Estimation of Tree Mixture Approximations

Sparse Graphical Model Selection: Intuitions

- First consider a graphical model with no latent variables

Markov Property of Graphical Models

$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S \iff I(X_u; X_v | \mathbf{X}_S) = 0$$

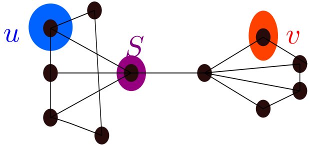


Sparse Graphical Model Selection: Intuitions

- First consider a graphical model with no latent variables

Markov Property of Graphical Models

$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S \iff I(X_u; X_v | \mathbf{X}_S) = 0$$



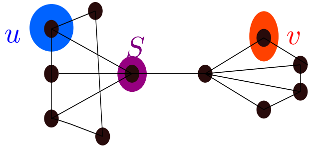
Alternative Test for Conditional Independence?

Sparse Graphical Model Selection: Intuitions

- First consider a graphical model with no latent variables

Markov Property of Graphical Models

$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S \iff I(X_u; X_v | \mathbf{X}_S) = 0$$



Alternative Test for Conditional Independence?

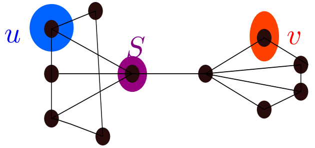
$$P(X_u = i, X_v = j | \mathbf{X}_S = k) = P(X_u = i | \mathbf{X}_S = k) \, P(X_v = j | \mathbf{X}_S = k)$$

Sparse Graphical Model Selection: Intuitions

- First consider a graphical model with no latent variables

Markov Property of Graphical Models

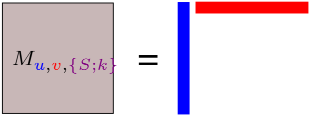
$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S \iff I(X_u; X_v | \mathbf{X}_S) = 0$$



Alternative Test for Conditional Independence?

$$P(X_u = i, X_v = j | \mathbf{X}_S = k) = P(X_u = i | \mathbf{X}_S = k) \, P(X_v = j | \mathbf{X}_S = k)$$

$$M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}.$$

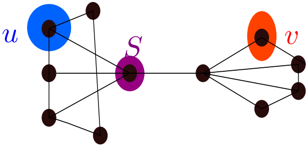


Sparse Graphical Model Selection: Intuitions

- First consider a graphical model with no latent variables

Markov Property of Graphical Models

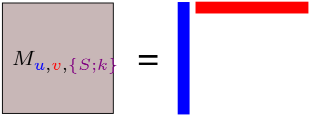
$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S \iff \text{Rank}(M_{u,v,\{S;k\}}) = 1$$



Alternative Test for Conditional Independence?

$$P(X_u = i, X_v = j | \mathbf{X}_S = k) = P(X_u = i | \mathbf{X}_S = k) \; P(X_v = j | \mathbf{X}_S = k)$$

$$M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}.$$

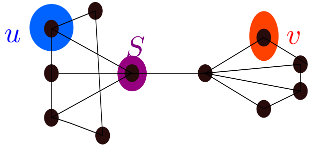


Sparse Graphical Model Selection: Intuitions

- First consider a graphical model with no latent variables

Markov Property of Graphical Models

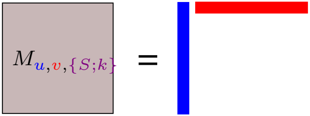
$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S \iff \text{Rank}(M_{u,v,\{S;k\}}) = 1$$



Alternative Test for Conditional Independence?

$$P(X_u = i, X_v = j | \mathbf{X}_S = k) = P(X_u = i | \mathbf{X}_S = k) \; P(X_v = j | \mathbf{X}_S = k)$$

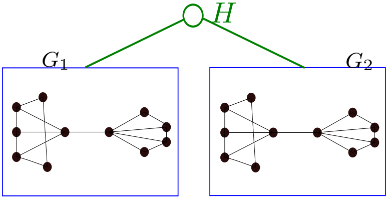
$$M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}.$$



Rank Test on Pairwise Probability Matrices

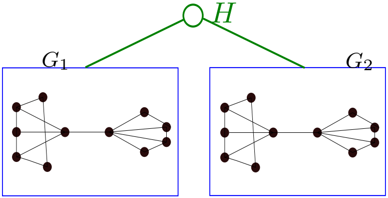
Extending Rank Tests to Mixtures

- Dimension of latent H is r and each observed variable is $d > r$.



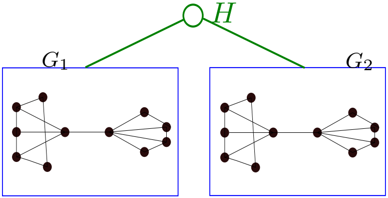
Extending Rank Tests to Mixtures

- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.

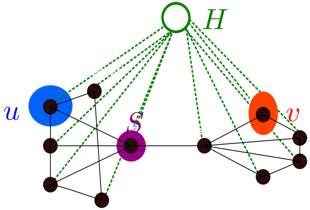


Extending Rank Tests to Mixtures

- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.

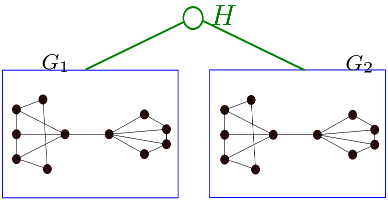


$X_u \not\perp X_v | \mathbf{X}_S$

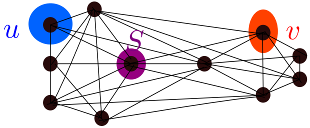


Extending Rank Tests to Mixtures

- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.

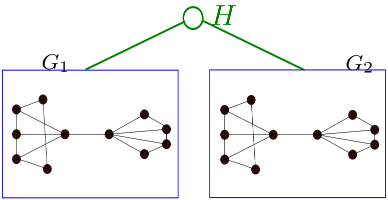


$X_u \not\perp\!\!\!\perp X_v | \mathbf{X}_S$

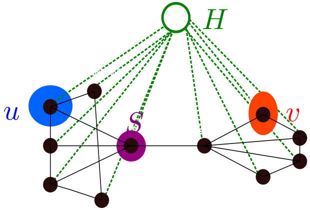


Extending Rank Tests to Mixtures

- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.

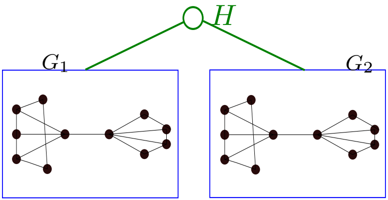


$$X_u \not\perp X_v | \mathbf{X}_S$$

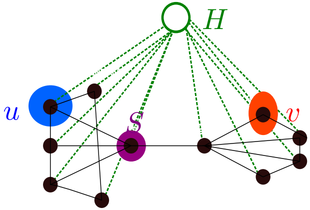


Extending Rank Tests to Mixtures

- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.

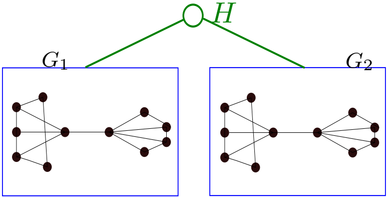


$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S, H$$

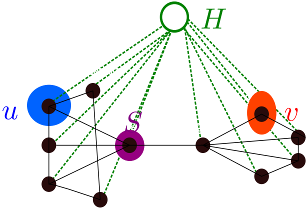


Extending Rank Tests to Mixtures

- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.



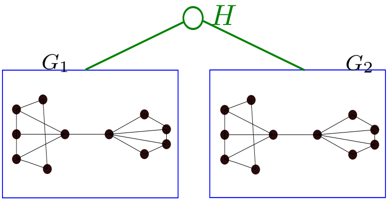
$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S, H$$



$$P(X_u, X_v|\mathbf{X}_S) = \sum_{h=1}^{r} \boxed{P(X_u|\mathbf{X}_S, \mathbf{H}=\mathbf{h})} \boxed{P(\mathbf{H}=\mathbf{h}|\mathbf{X}_S)} \boxed{P(X_v|\mathbf{X}_S, \mathbf{H}=\mathbf{h})}$$

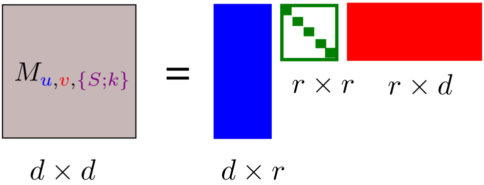
Extending Rank Tests to Mixtures

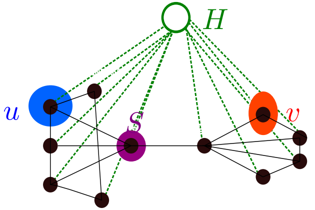
- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.



$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S, H$$

$$M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}.$$

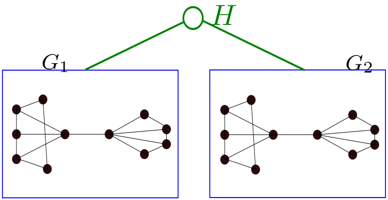




$$P(X_u, X_v | \mathbf{X}_S) = \sum_{h=1}^{r} \boxed{P(X_u | \mathbf{X}_S, \mathbf{H} = \mathbf{h})} \; \boxed{P(\mathbf{H} = \mathbf{h} | \mathbf{X}_S)} \; \boxed{P(X_v | \mathbf{X}_S, \mathbf{H} = \mathbf{h})}$$

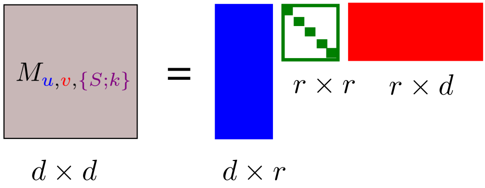
Extending Rank Tests to Mixtures

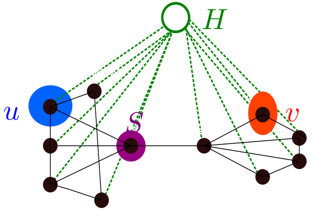
- Dimension of latent H is r and each observed variable is $d > r$.
- First assume Markov graph is the same for all components.



$$X_u \perp\!\!\!\perp X_v | \mathbf{X}_S, H \iff \mathrm{Rank}(M_{u,v,\{S;k\}}) = r$$

$M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}$.

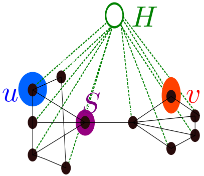




$$P(X_u, X_v | \mathbf{X}_S) = \sum_{h=1}^{r} \boxed{P(X_u | \mathbf{X}_S, \mathbf{H} = \mathbf{h})} \; \boxed{P(\mathbf{H} = \mathbf{h} | \mathbf{X}_S)} \; \boxed{P(X_v | \mathbf{X}_S, \mathbf{H} = \mathbf{h})}$$

Rank Test for Mixtures

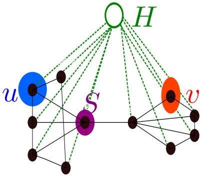
- Dim(H) is r and each observed variable is $d > r$.
- $G_\cup$: union of Markov graphs of components.
- η: Bound on separators btw. node pairs in $G_\cup$.
- $M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}$



Declare (u, v) as edge if $\min_{\substack{S \subset V \setminus \{u,v\} \\ |S| \leq \eta}} \max_{k \in \mathcal{X}^{|S|}} \text{Rank}(M_{u,v,\{S;k\}}; \xi_{n,p}) > r.$

Rank Test for Mixtures

- $\text{Dim}(H)$ is r and each observed variable is $d > r$.
- $G_\cup$: union of Markov graphs of components.
- η: Bound on separators btw. node pairs in $G_\cup$.
- $M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}$

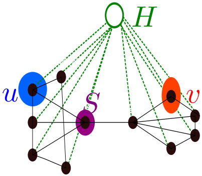


Declare (u, v) as edge if $\min_{\substack{S \subset V \setminus \{u,v\} \\ |S| \leq \eta}} \max_{k \in \mathcal{X}^{|S|}} \text{Rank}(M_{u,v,\{S;k\}}; \xi_{n,p}) > r.$

Small $\eta \Rightarrow$ computationally efficient, uses only low order statistics.

Rank Test for Mixtures

- Dim(H) is r and each observed variable is $d > r$.
- $G_\cup$: union of Markov graphs of components.
- η: Bound on separators btw. node pairs in $G_\cup$.
- $M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}$



Declare (u, v) as edge if $\min_{\substack{S \subset V \setminus \{u,v\} \\ |S| \leq \eta}} \max_{k \in \mathcal{X}^{|S|}} \mathrm{Rank}(M_{u,v,\{S;k\}}; \xi_{n,p}) > r$.

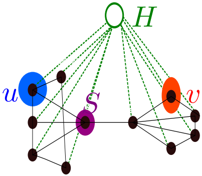
Small $\eta \Rightarrow$ computationally efficient, uses only low order statistics.

Examples of graphs $G_\cup$ with small η

- Mixture of product distributions: $G_\cup$ is trivial and $\eta = 0$.
- Mixture on same tree: $G_\cup$ is a tree and $\eta = 1$.
- Mixture on arbitrary trees: $G_\cup$ is union of r trees and $\eta = r$.

Rank Test for Mixtures

- $\text{Dim}(H)$ is r and each observed variable is $d > r$.
- $G_\cup$: union of Markov graphs of components.
- η: Bound on separators btw. node pairs in $G_\cup$.
- $M_{u,v,\{S;k\}} := [P(X_u = i, X_v = j, \mathbf{X}_S = k)]_{i,j}$



Declare (u, v) as edge if $\min_{\substack{S \subset V \setminus \{u,v\} \\ |S| \leq \eta}} \max_{k \in \mathcal{X}^{|S|}} \text{Rank}(M_{u,v,\{S;k\}}; \xi_{n,p}) > r.$

Small $\eta \Rightarrow$ computationally efficient, uses only low order statistics.

Examples of graphs $G_\cup$ with small η

- Mixture of product distributions: $G_\cup$ is trivial and $\eta = 0$.
- Mixture on same tree: $G_\cup$ is a tree and $\eta = 1$.
- Mixture on arbitrary trees: $G_\cup$ is union of r trees and $\eta = r$.

Guarantees on Rank Test

Theorem (**A.** , Hsu, Huang, Kakade '12)

Rank test recovers graph structure $G_\cup$ correctly w.h.p on p nodes under n samples when

$$\boxed{\frac{\rho_{\min}^{-2}\log p}{n} = O(1).}$$

- $\rho_{\min}$: Min. $(r+1)^{\text{th}}$ singular value between neighbors.

Guarantees on Rank Test

Theorem (**A.** , Hsu, Huang, Kakade '12)

Rank test recovers graph structure $G_\cup$ correctly w.h.p on p nodes under n samples when

$$\frac{\rho_{\min}^{-2} \log p}{n} = O(1).$$

- $\rho_{\min}$: Min. $(r+1)^{\text{th}}$ singular value between neighbors.

Efficient Test with Low Sample and Computational Requirements.

Guarantees on Rank Test

Theorem (**A.** , Hsu, Huang, Kakade '12)

Rank test recovers graph structure $G_\cup$ correctly w.h.p on p nodes under n samples when

$$\boxed{\frac{\rho_{\min}^{-2} \log p}{n} = O(1).}$$

- $\rho_{\min}$: Min. $(r+1)^{\text{th}}$ singular value between neighbors.

Efficient Test with Low Sample and Computational Requirements.

Recall examples of graphs $G_\cup$ with small η

- Mixture of product distributions: $G_\cup$ is trivial and $\eta = 0$.
- Mixture on same tree: $G_\cup$ is a tree and $\eta = 1$.
- Mixture on arbitrary trees: $G_\cup$ is union of r trees and $\eta = r$.

Guarantees for Learning Graphical Model Mixtures

Steps Involved in Tree Mixture Approximation

- Rank tests for structure estimation of union graph $G_\cup$
- Tensor decomposition for estimation of pairwise moments of mixture components
- Chow-Liu algorithm to estimate mixture component trees

Computationally Efficient Algorithm for Learning Graphical Model Mixtures

---

"Learning High-Dimensional Mixtures of Graphical Models" by A. Anandkumar, D. Hsu, F. Huang, and S.M. Kakade. NIPS 2012.

Guarantees for Learning Graphical Model Mixtures

Steps Involved in Tree Mixture Approximation

- Rank tests for structure estimation of union graph $G_\cup$
- Tensor decomposition for estimation of pairwise moments of mixture components
- Chow-Liu algorithm to estimate mixture component trees

Computationally Efficient Algorithm for Learning Graphical Model Mixtures

Theorem (A. , Hsu, Huang, Kakade '12)

The above method recovers correct tree mixture approximation correctly w.h.p on p nodes of r component mixture under n samples when

$$\boxed{n = \mathrm{poly}(p, r).}$$

"Learning High-Dimensional Mixtures of Graphical Models" by A. Anandkumar, D. Hsu, F. Huang, and S.M. Kakade. NIPS 2012.

Guarantees for Learning Graphical Model Mixtures

Steps Involved in Tree Mixture Approximation

- Rank tests for structure estimation of union graph $G_\cup$
- Tensor decomposition for estimation of pairwise moments of mixture components
- Chow-Liu algorithm to estimate mixture component trees

Computationally Efficient Algorithm for Learning Graphical Model Mixtures

Theorem (A. , Hsu, Huang, Kakade '12)

The above method recovers correct tree mixture approximation correctly w.h.p on p nodes of r component mixture under n samples when

$$\boxed{n = \mathrm{poly}(p, r).}$$

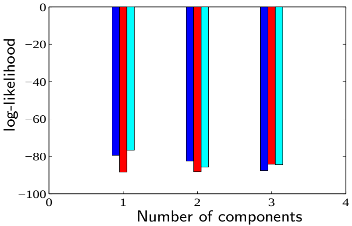
Efficient Learning of Multiple Graphs and Models in High Dimensions

---

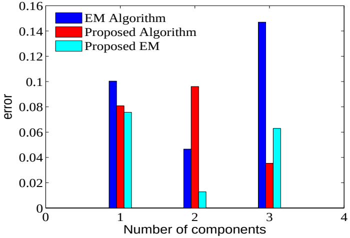
"Learning High-Dimensional Mixtures of Graphical Models" by A. Anandkumar, D. Hsu, F. Huang, and S.M. Kakade. NIPS 2012.

Splice Data

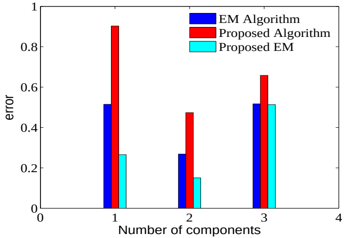
- DNA SPLICE-junctions
- 60 variables(sequence of DNA bases) , class variable
- Splice junction type: EI, IE, none.



Conditional log-likelihood



Mixing weight estimation(r=3)



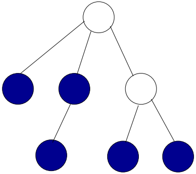
Classification Error

Outline

Summary and Outlook

Learning Latent Tree Models

- Integrated Structure and Parameter Learning
- High level of parallelism without losing consistency.



Learning Graphical Model Mixtures

- Tree mixture approximations
- Combinatorial search + spectral decomposition
- Computational and sample guarantees