10-701 Introduction to Machine Learning, Course Project
keywords: MCMC, Gibbs Sampling, Markov Networks, Mixing Time

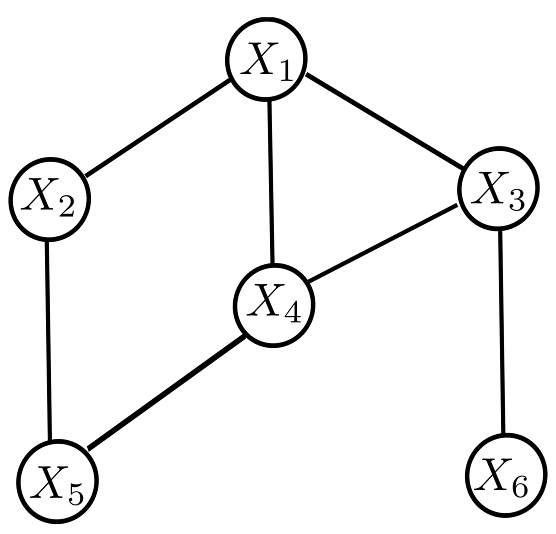
Improving Mixing times in Gibbs Sampling

Wenlu Hu Kirthevasan Kandasamy Joseph Tassarotti
{wenluh, kkandasa, jtassaro}@cs.cmu.edu

Summary

Improve mixing time in Gibbs Sampling via Tree Blocks that group correlated variables together.

Introduction



Given: Graphical Model with known Parameters.

Goal: Estimate $\mathbb{E}[h(X)]$
E.g.: $1_A(x)$, $\sum_i X_i$

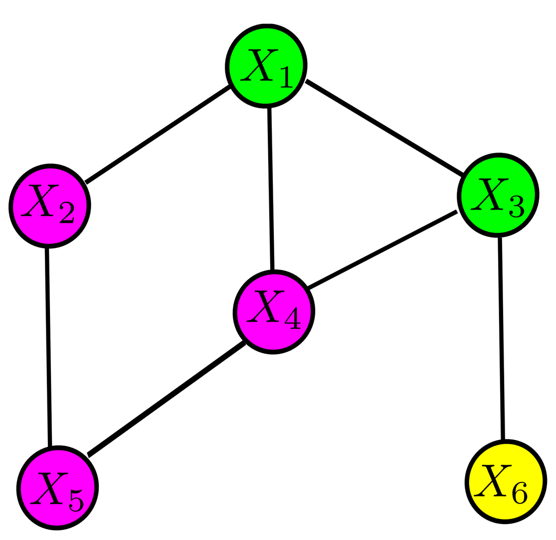
Generate Samples: $X^{(1)}, X^{(2)}, \dots X^{(N)}$

Empirical Estimator

$$\mu_0 = \frac{1}{N}\sum_{i=1}^{N} h(X^{(i)})$$

Gibbs Sampling–one way to generate samples.

Blocked Gibbs Sampling



Current Sample: $X^{(t)}$

$X_{1,3}^{(t+1)} \mid X_{2,4,5,6}^{(t)}$

$X_{2,4,5}^{(t+1)} \mid X_{1,3}^{(t+1)}, X_6^{(t)}$

$X_6^{(t+1)} \mid X_{1,2,3,4,5}^{(t+1)}$

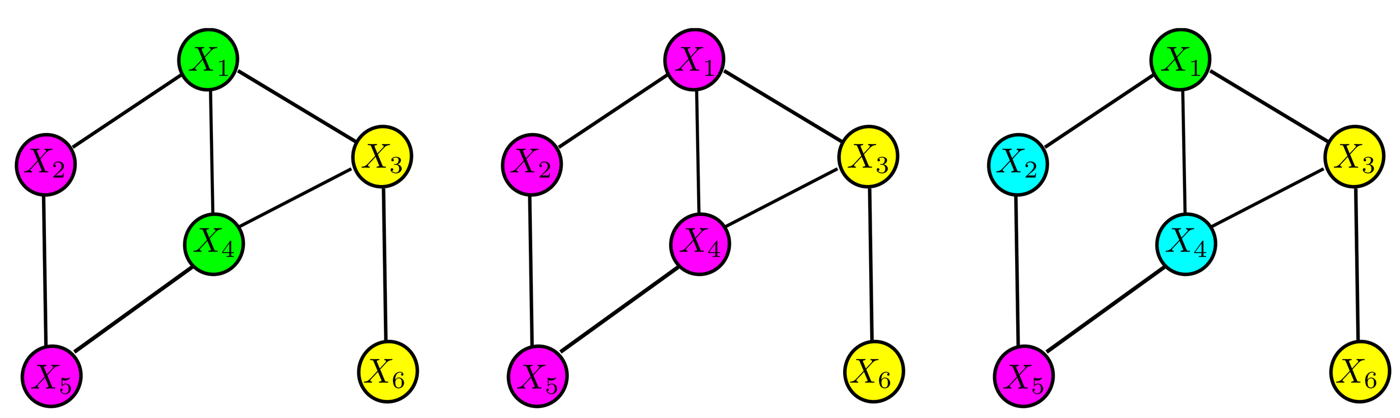
Next Sample: $X^{(t+1)}$

Why ?

Sampling is now more difficult.

BUT, Chain mixes faster $\implies$ better samples.

How to Block ?



Strategy

- We will focus only on **Tree Partitions**.
  - Otherwise Problem is too big.
  - Inference on Trees is easy (Belief Propagation converges in linear time).
- Will consider **correlations** between variables when developing tree blocks.

Why Correlations ?

MC: $X^{(1)} \to X^{(2)} \to \dots \to X^{(t)} \to X^{(t+1)} \to \dots,$ Eqlbm Distribtuion: π

$L_0(\pi) = \{h : \Omega \to \mathbb{R} : \ \mathbb{E}_\pi h(X) = 0, \ \mathbb{V}_\pi h(X) < \infty\}$,
$\langle h, g\rangle = \mathrm{Covar}_\pi(h(X), g(X))$
$\big(L_0(\pi), \langle\cdot,\cdot\rangle\big)$ is a Hilbert Space.

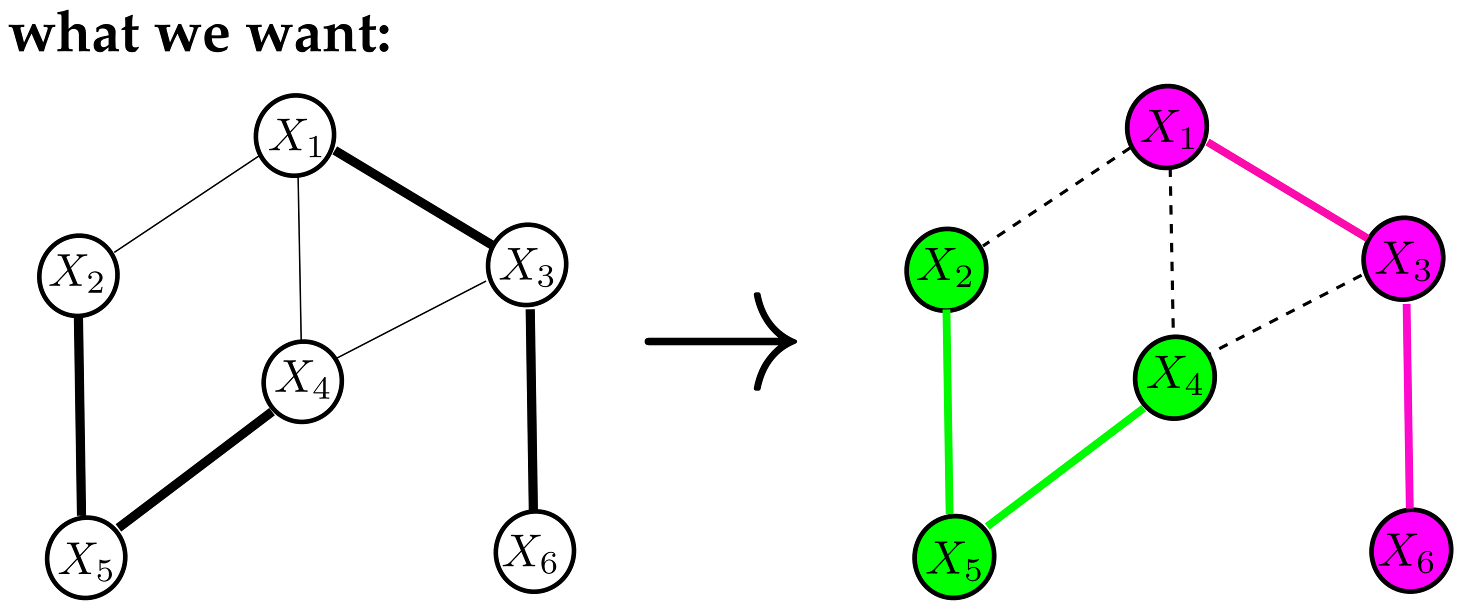
Define $F : L_0(\pi) \to L_0(\pi)$, $[Fh](z) = \mathbb{E}[h(X^{(1)}) | X^{(0)} = z]$

Fact: $|\mathbb{E}^{(n)} h(X) - \mathbb{E}_\pi h(X)| \leqslant C \, \|F\|^n \, \|h\|$

$\|F\| = \sup_{f,g} \mathrm{Corr}(f(X^{(t+1)}), g(X^{(t)}))$

Correlated Variables in different blocks $\implies$ successive samples correlated.

So this is what we want:

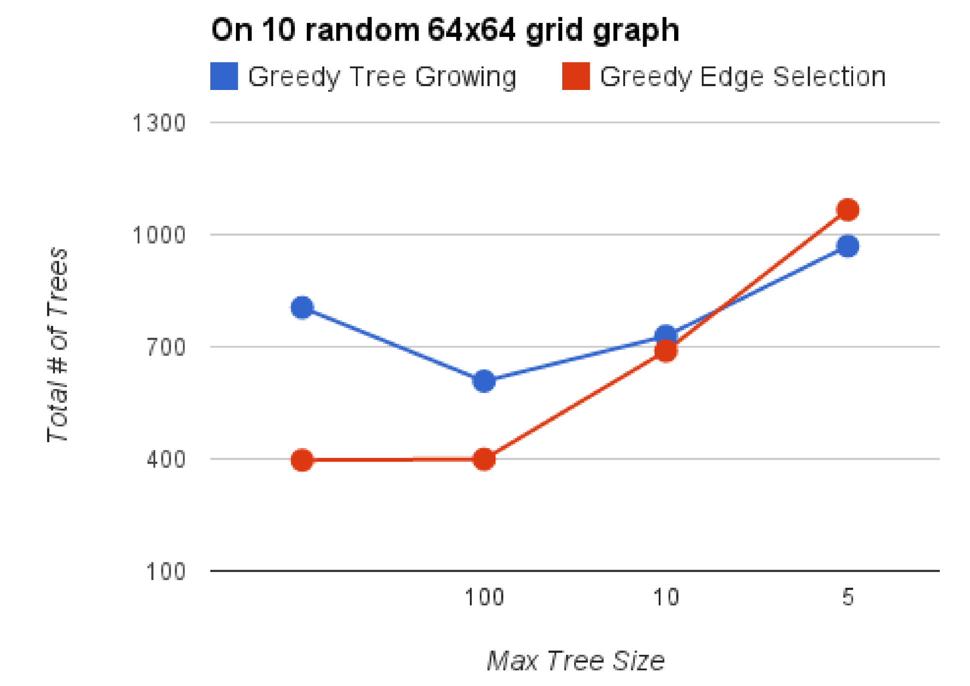


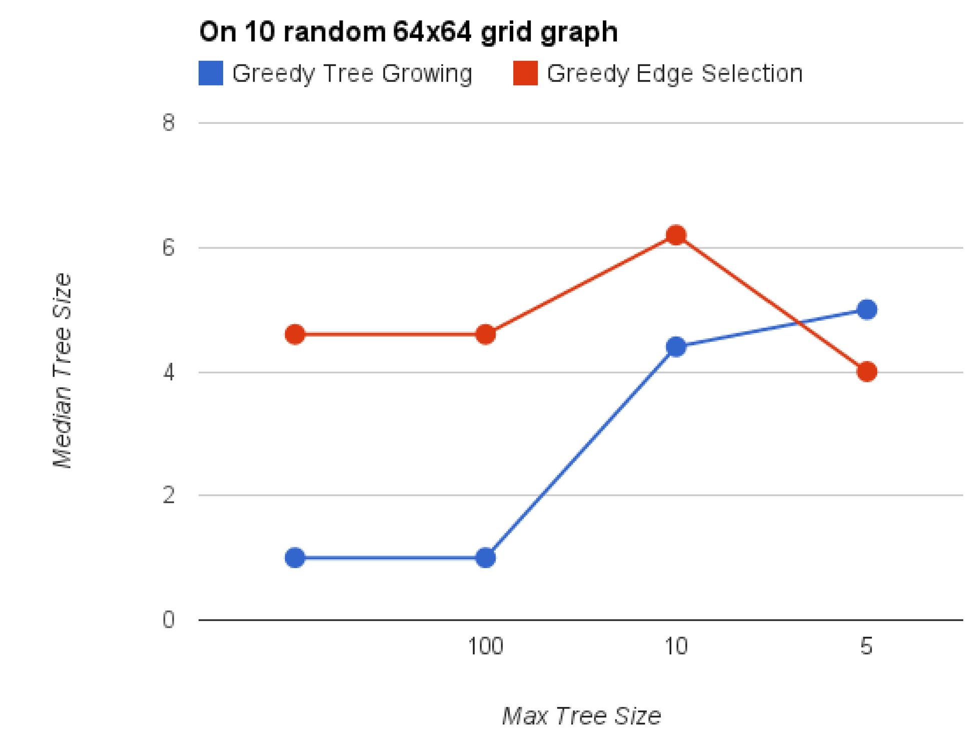
Algorithms

- Baseline: Greedy Tree Growing Algorithm
- Our Algorithm: Greedy Edge Selection Algorithm

Greedy Edge Selection Algorithm

1. Construct an ordered list of edges, E, with $E[0]$ being the highest weight edge. Edges are vertex pairs (i, j).
2. Initialize an all-zero n-dimensional integer list V of vertex colors.
   ($V[i]$ is the color of vertex i, and $V[i] = 0$ means that vertex i has not yet been colored.)
3. Initialize n empty vertex sets: $T_1, \dots, T_n$
   (Logically, T_i is the set of vertices labeled with color i.)
4. Initialize $\mathit{unusedColor} = 1$.
5. For each edge $e = (i, j)$ in E,
   - If $V[i] = V[j] = 0$,
     - Set $V[i] = V[j] = \mathit{unusedColor}$
     - Add i, j to $T_{\mathit{unusedColor}}$
     - Increment $\mathit{unusedColor}$ by 1
   - Else if $V[i] = 0$ and $V[j] \notin \mathit{getOtherNeighborColors}(\{i\}, e)$,
     - Set $V[i] = V[j]$
     - Add i to $T_{V[j]}$
   - Else if $V[j] = 0$ and $V[i] \notin \mathit{getOtherNeighborColors}(\{j\}, e)$,
     - Set $V[j] = V[i]$
     - Add j to $T_{V[i]}$
   - Else if $V[i] \neq 0$ and $V[j] \neq 0$ and $V[i] \notin \mathit{getOtherNeighborColors}(T_j, e)$,
     - For each $k \in T_j$, set $V[k] = V[i]$
     - Set $T_i = T_i \cup T_j$
     - Set $T_j = \emptyset$
   - Otherwise do nothing
6. For each vertex i, if $V[i] = 0$, set $V[i] = \mathit{unusedColor}, \mathit{unusedColor}++$
7. Output $\{T_j : T_j \neq \emptyset\}$





Greedy Tree Growing Algorithm

1. Initialize $i = 0$ and V to the vertex set.
2. While $V \neq \emptyset$
   - Select $v \in V$
   - Start a new tree T_i and a priority queue Q_i. Add v to Q_i
   - While $Q_i \neq \emptyset$
     - Pop u from Q_i.
     - Initialize $\mathit{neighborsInT} = 0$.
     - For all $v \in T_i$, if $u \in N(v)$, increment $\mathit{neighborsInT}$
     - If $\mathit{neighborsInT} \leqslant 1$,
     - Add u to T_i and remove v from V
     - Add $N(u)$ to Q_i.
3. Return $\{T_i\}$

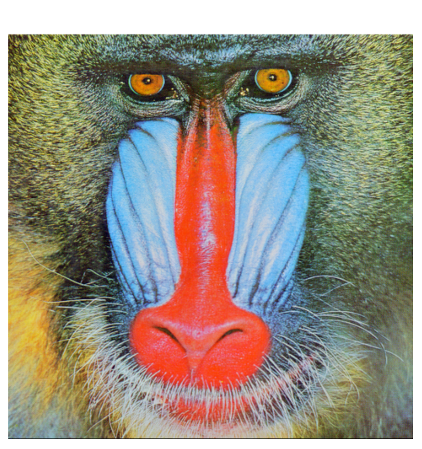
Greedy Edge Selection Algorithm performs much better when max tree size is not limited.

As max tree size is controlled, Greedy Tree Growing Algorithm caught up.

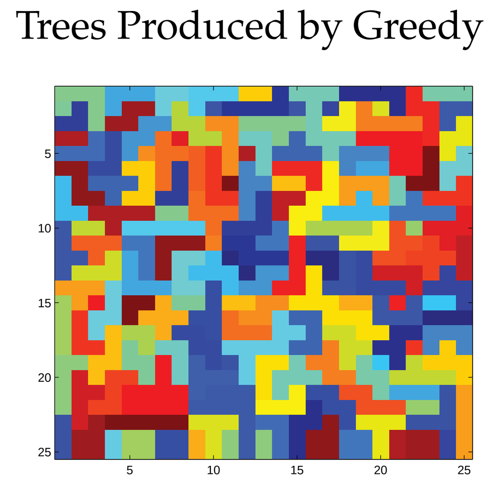
Experiments - Image Reconstruction

Estimated Correlations by Running a Gibbs Sampler – **Cheating !**
This information could come via Expert Knowledge, Repetitive tasks etc.

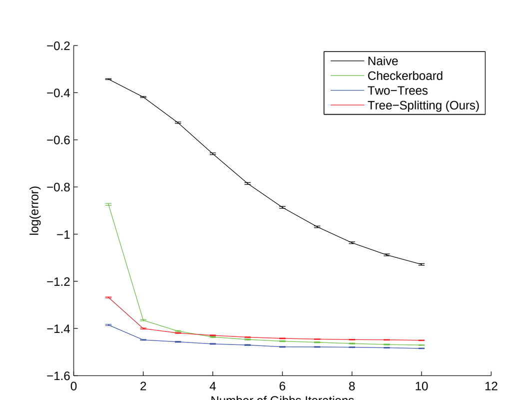
Image



Trees Produced by Greedy Edge



Error vs # Iterations



Note that CB/ Two-Trees have only 2 trees whereas we had > 200 trees.

Conclusion

- Proposed method has good convergence in number of iterations.
- But Naive Gibbs still beats us in computation time $\implies$ need better tree sampling implementation.
- Criticism: Neighboring edges in a UGM are already quite correlated. So unsure how well this would work in practice.

Acknowledgements

Elara Willet for helping us with the Graph Theory element of the project.

References :

Firas Hamze, Nando de Freitas, **From Fields to Trees**, UAI 2004
Liu Jun S, **Markov Chain Strategies in Scientific Computing**, 2001
Liu Jun S, Wong Wing H, Kong Augustine, **Covariance structure of the Gibbs Sampler with applications to the comparisons of estimators and augmentation schemes**, Biometrika 1994.