Inference in Chains and Trees

Mark Schmidt

University of British Columbia

August, 2015

Notation from Last Time

- We're focusing on pairwise UGMs with discrete states,

$$P(X) = \frac{\prod_{i=1}^{N} \phi_i(x_i) \prod_{(i,j) \in E} \phi_{ij}(x_i, x_j)}{Z},$$

where we've decomposed object X into 'parts' $x_i \in \{1, 2, \dots, S\}$.

Notation from Last Time

- We're focusing on pairwise UGMs with discrete states,

$$P(X) = \frac{\prod_{i=1}^{N} \phi_i(x_i) \prod_{(i,j) \in E} \phi_{ij}(x_i, x_j)}{Z},$$

  where we've decomposed object X into 'parts' $x_i \in \{1, 2, \dots, S\}$.

- Week 1 considers exact methods for 3 tasks:
  1. Decoding: Compute the optimal configuration,
  $$\max_X P(X).$$
  2. Inference: Compute partition function and marginals,
  $$Z = \sum_{X'} P(X'), \quad P(X_i = s) = \sum_{X' \mid X_i = s} p(X').$$
  3. Sampling: Generate X' according to Gibbs distribution:
  $$X' \sim P(X).$$

Computer Science Graduate Markov Model

- Computer Science Graduate Careers Markov chain:
  - Variable x_1 can be in one of three states:

State	Probability	Description
Industry	0.60	They work for a company or own their own company.
Grad School	0.30	They are trying to get a Masters or PhD degree.
Video Games	0.10	They mostly play video games.

Computer Science Graduate Markov Model

- Computer Science Graduate Careers Markov chain:
  - Variable x_1 can be in one of three states:

State	Probability	Description
Industry	0.60	They work for a company or own their own company.
Grad School	0.30	They are trying to get a Masters or PhD degree.
Video Games	0.10	They mostly play video games.

  - Variable x_t only depends on x_{t-1}:

From\to	Video Games	Industry	Grad School	Video Games (with PhD)	Industry (with PhD)	Academia	Deceased
Video Games	0.08	0.90	0.01	0	0	0	0.01
Industry	0.03	0.95	0.01	0	0	0	0.01
Grad School	0.06	0.06	0.75	0.05	0.05	0.02	0.01
Video Games (with PhD)	0	0	0	0.30	0.60	0.09	0.01
Industry (with PhD)	0	0	0	0.02	0.95	0.02	0.01
Academia	0	0	0	0.01	0.01	0.97	0.01
Deceased	0	0	0	0	0	0	1

Computer Science Graduate Markov Model

- Computer Science Graduate Careers Markov chain:
  - Variable x_1 can be in one of three states:

State	Probability	Description
Industry	0.60	They work for a company or own their own company.
Grad School	0.30	They are trying to get a Masters or PhD degree.
Video Games	0.10	They mostly play video games.

  - Variable x_t only depends on x_{t-1}:

From\to	Video Games	Industry	Grad School	Video Games (with PhD)	Industry (with PhD)	Academia	Deceased
Video Games	0.08	0.90	0.01	0	0	0	0.01
Industry	0.03	0.95	0.01	0	0	0	0.01
Grad School	0.06	0.06	0.75	0.05	0.05	0.02	0.01
Video Games (with PhD)	0	0	0	0.30	0.60	0.09	0.01
Industry (with PhD)	0	0	0	0.02	0.95	0.02	0.01
Academia	0	0	0	0.01	0.01	0.97	0.01
Deceased	0	0	0	0	0	0	1

- So the probability of a sequence is

$$\begin{aligned} p(x_1, x_2, \dots, x_n) &= p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)\dots p(x_n|x_{n-1}, x_{n-2}, \dots, x_1) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_2)\dots p(x_n|x_{n-1}). \end{aligned}$$

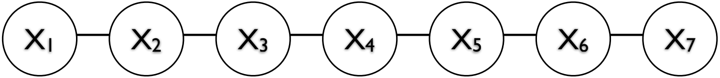
- Markov property: $p(x_j|x_{j-1}, x_{j-2}, \dots, x_1) = p(x_j|x_{j-1})$.

Markov Chain Models

- This is a special case of a UGM

$$p(x_1, x_2, \ldots, x_n) \propto \phi_1(x_1) \prod_{i=2}^{n} \phi(x_i, x_{i-1}),$$

with a chain-structured dependency:



- Homogeneous chain: edge potentials are constant across time.
- Markov chains are ubiquitous in sequence/time-series models:

9 Applications
- 9.1 Physics
- 9.2 Chemistry
- 9.3 Testing
- 9.4 Speech Recognition
- 9.5 Information sciences
- 9.6 Queueing theory
- 9.7 Internet applications
- 9.8 Statistics
- 9.9 Economics and finance
- 9.10 Social sciences
- 9.11 Mathematical biology
- 9.12 Genetics
- 9.13 Games
- 9.14 Music
- 9.15 Baseball
- 9.16 Markov text generators

General Chain-Structured UGM

- The general class of chain-structured UGMs is

$$p(x_1, x_2, \dots, x_n) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{i=2}^{n} \phi_{i,i-1}(x_i, x_{i-1}),$$

  (x_t could depend on future things that might happen)
- In this case we only have local Markov property,

$$x_i \perp x_1, \dots, x_{i-2}, x_{i+2}, \dots, x_n | x_{i-1}, x_{i+1},$$

General Chain-Structured UGM

- The general class of chain-structured UGMs is

$$p(x_1, x_2, \ldots, x_n) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{i=2}^{n} \phi_{i,i-1}(x_i, x_{i-1}),$$

  (x_t could depend on future things that might happen)
- In this case we only have local Markov property,

$$x_i \perp x_1, \ldots, x_{i-2}, x_{i+2}, \ldots, x_n | x_{i-1}, x_{i+1},$$

- Local Markov property in general UGMs:
  - given neighbours, conditional independence of other nodes.

General Chain-Structured UGM

- The general class of chain-structured UGMs is

$$p(x_1, x_2, \dots, x_n) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{i=2}^{n} \phi_{i,i-1}(x_i, x_{i-1}),$$

  (x_t could depend on future things that might happen)

- In this case we only have local Markov property,

$$x_i \perp x_1, \dots, x_{i-2}, x_{i+2}, \dots, x_n | x_{i-1}, x_{i+1},$$

- Local Markov property in general UGMs:
  - given neighbours, conditional independence of other nodes.

(Marginal independence corresponds to reachability.)

General Chain-Structured UGM

- The general class of chain-structured UGMs is

$$p(x_1, x_2, \ldots, x_n) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{i=2}^{n} \phi_{i,i-1}(x_i, x_{i-1}),$$

(x_t could depend on future things that might happen)

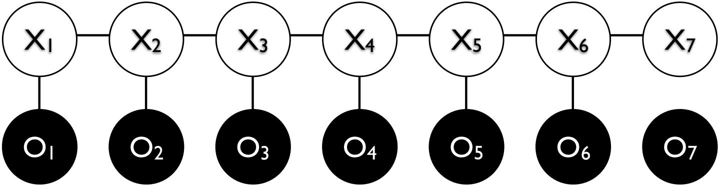
- In this case we only have local Markov property,

$$x_i \perp x_1, \ldots, x_{i-2}, x_{i+2}, \ldots, x_n | x_{i-1}, x_{i+1},$$

- Local Markov property in general UGMs:
  - given neighbours, conditional independence of other nodes.

(Marginal independence corresponds to reachability.)

- Includes hidden Markov models (HMMs) and Kalman filters:



Applications of HMMs and Kalman Filters

Applications [edit]

HMMs can be applied in many fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are). Applications include:

- Single Molecule Kinetic analysis[16]
- Cryptanalysis
- Speech recognition
- Speech synthesis
- Part-of-speech tagging
- Document Separation in scanning solutions
- Machine translation
- Partial discharge
- Gene prediction
- Alignment of bio-sequences
- Time Series Analysis
- Activity recognition
- Protein folding[17]
- Metamorphic Virus Detection[18]
- DNA Motif Discovery[19]

Applications [edit]

- Attitude and Heading Reference Systems
- Autopilot
- Battery state of charge (SoC) estimation[39][40]
- Brain-computer interface
- Chaotic signals
- Tracking and Vertex Fitting of charged particles in Particle Detectors[41]
- Tracking of objects in computer vision
- Dynamic positioning
- Economics, in particular macroeconomics, time series analysis, and econometrics[42]
- Inertial guidance system
- Orbit Determination
- Power system state estimation
- Radar tracker
- Satellite navigation systems
- Seismology[43]
- Sensorless control of AC motor variable-frequency drives
- Simultaneous localization and mapping
- Speech enhancement
- Visual odometry
- Weather forecasting
- Navigation system
- 3D modeling
- Structural health monitoring
- Human sensorimotor processing[44]

Also includes conditional random fields.

Cost of Decoding

- Last time and in homework, exact inference by table:

Cathy	Heather	Mark	Allison	np(1)	np(2)	np(3)	np(4)	ep(1)	ep(2)	ep(3)	prodPot	Probability
right	right	right	right	1	9	1	9	2	2	2	648	0.17
wrong	right	right	right	3	9	1	9	1	2	2	972	0.26
right	wrong	right	right	1	1	1	9	1	1	2	18	0.00
wrong	wrong	right	right	3	1	1	9	2	1	2	108	0.03
right	right	wrong	right	1	9	3	9	2	1	1	486	0.13
wrong	right	wrong	right	3	9	3	9	1	1	1	729	0.19
right	wrong	wrong	right	1	1	3	9	1	2	1	54	0.01
wrong	wrong	wrong	right	3	1	3	9	2	2	1	324	0.09
right	right	right	wrong	1	9	1	1	2	2	1	36	0.01
wrong	right	right	wrong	3	9	1	1	1	2	1	54	0.01
right	wrong	right	wrong	1	1	1	1	1	1	1	1	0.00
wrong	wrong	right	wrong	3	1	1	1	2	1	1	6	0.00
right	right	wrong	wrong	1	9	3	1	2	1	2	108	0.03
wrong	right	wrong	wrong	3	9	3	1	1	1	2	162	0.04
right	wrong	wrong	wrong	1	1	3	1	1	2	2	12	0.00
wrong	wrong	wrong	wrong	3	1	3	1	2	2	2	72	0.02

Decoding in Chain-Structured Models

- Table is too expensive for Markov chain models:
  - We can't enumerate s^n possible configurations.

Decoding in Chain-Structured Models

- Table is too expensive for Markov chain models:
  - We can't enumerate s^n possible configurations.
- To avoid this use Markov property and dynamic programming:
  - Assume you know optimal value at time t.
  - By Markov property, captures everything about the past.
  - Use this to compute optimal value at time $t+1$.

Decoding in Chain-Structured Models

- Viterbi decoding algorithm:
  - Forward phase:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \max_{s'}\{\phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}\},$$

  - Backward phase: backtrack through argmax values.
  - Solves the decoding problem in $O(ns^2)$ instead of $O(s^n)$.

Decoding in Chain-Structured Models

- Viterbi decoding algorithm:
  - Forward phase:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \max_{s'}\{\phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}\},$$

  - Backward phase: backtrack through argmax values.
  - Solves the decoding problem in $O(ns^2)$ instead of $O(s^n)$.
- For the CS grad student Markov model with $n = 60$:
  - Optimal decoding is ‘industry’ for each year.

Decoding in Chain-Structured Models

- Viterbi decoding algorithm:
  - Forward phase:

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \max_{s'}\{\phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}\},$$

  - Backward phase: backtrack through argmax values.
  - Solves the decoding problem in $O(ns^2)$ instead of $O(s^n)$.
- For the CS grad student Markov model with $n = 60$:
  - Optimal decoding is ‘industry’ for each year.
  - Optimal decoding might not look like ‘typical’ state.
  - Optimal decoding would be different with inhomogeneous chain.
  - Optimal decoding would be different if we changed n.

Inference in Chain-Structured Models

- Chapman-Kolmogorov equations for inference in Markov chains:
  - Dynamic programming to sum up all paths to state s at time t,

$$V_{1,s} = p(s), \quad V_{i,s} = \sum_{s'} p(s|s')V_{i-1,s'}, \quad Z = \sum_{s} V_{n,s},$$

    and get marginal $p(x_i = s)$ by normalizing $V_{i,s}$ across s.

Inference in Chain-Structured Models

- Chapman-Kolmogorov equations for inference in Markov chains:
  - Dynamic programming to sum up all paths to state s at time t,

    $$V_{1,s} = p(s), \quad V_{i,s} = \sum_{s'} p(s|s')V_{i-1,s'}, \quad Z = \sum_{s} V_{n,s},$$

    and get marginal $p(x_i = s)$ by normalizing $V_{i,s}$ across s.
  - Needs marginals/conditionals: can't apply to general chain-structured UGMs.

Inference in Chain-Structured Models

- Chapman-Kolmogorov equations for inference in Markov chains:
  - Dynamic programming to sum up all paths to state s at time t,

$$V_{1,s} = p(s), \quad V_{i,s} = \sum_{s'} p(s|s')V_{i-1,s'}, \quad Z = \sum_{s} V_{n,s},$$

    and get marginal $p(x_i = s)$ by normalizing $V_{i,s}$ across s.
  - Needs marginals/conditionals: can't apply to general chain-structured UGMs.
- Forward-backward algorithm for general case:
  - Forward phase (sums up paths from the beginning):

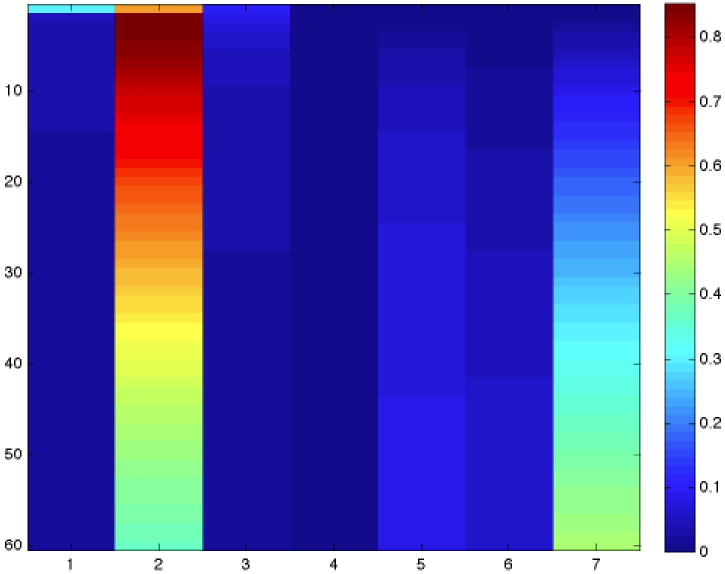
$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \sum_{s'} \phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}, \quad Z = \sum_{s} V_{n,s}.$$

  - Backward phase: (sums up paths to the end):

$$B_{n,s} = 1, \quad B_{i,s} = \sum_{s'} \phi_{i+1}(s')\phi_{i+1,i}(s',s)B_{i+1,s'}.$$

  - Marginals are given by $p(x_i = s) \propto V_{i,s}B_{i,s}$.

Marginals in CS Grad Markov Chain



Sampling in Chain-Structured Models

- Sampling is easy in Markov chains:
  - Sample time 1 based on $p(x_1)$.
  - Sample time t based on time $t-1$ using $p(x_t|x_{t-1})$.
  - Simulates the process forward from the beginning.

Sampling in Chain-Structured Models

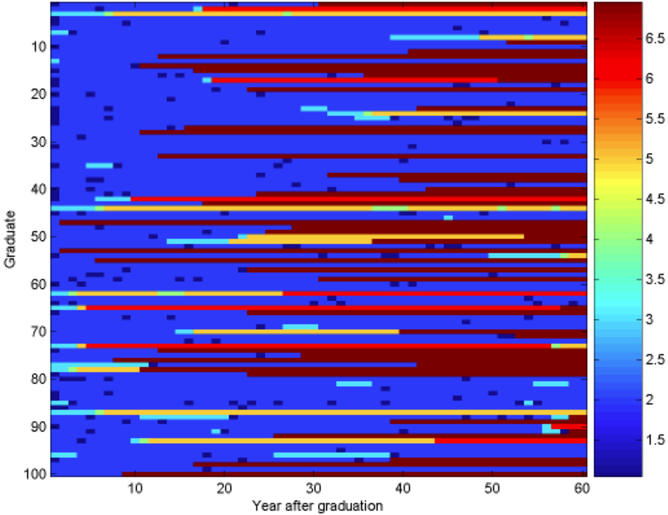
- Sampling is easy in Markov chains:
  - Sample time 1 based on $p(x_1)$.
  - Sample time t based on time $t-1$ using $p(x_t|x_{t-1})$.
  - Simulates the process forward from the beginning.
- Forward-filter backward-sample algorithm for general case:
  - Forward phase (same as before):

$$V_{1,s} = \phi_1(s), \quad V_{i,s} = \sum_{s'} \phi_i(s)\phi_{i,i-1}(s,s')V_{i-1,s'}.$$

  - Backward phase: sample x_n now that we have $p(x_n)$, then sample time $(t-1)$ based on $V_{t-1,s}$ and x_t.
  - Simulates the process backwards from the end.

Samples in CS Grad Markov Chain

Samples are more informative about what the model looks like:



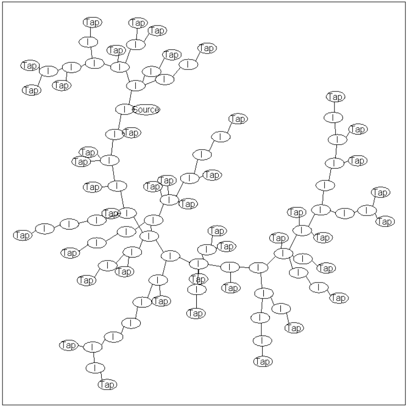
Could use samples to guide refining model.

Tree-Structured UGMs

- Decoding/inference/sampling in chains is $O(ns^2)$.

Tree-Structured UGMs

- Decoding/inference/sampling in chains is $O(ns^2)$.
- We can get the same runtime for trees (graph with no loops)



Forward phase idea: start from the leaves and work your way in.

- Call belief propagation, special case of message passing.

Belief Propagation

- For decoding ("max-product"), message from j to i has the form

$$m_{ji}(x_i) = \max_{x_j} \left\{ \phi_j(x_j)\phi_{i,j}(x_i, x_j) \prod_{k \in N(j)\setminus i} m_{kj}(x_j) \right\}.$$

- For inference ("sum-product"), message from j to i has the form

$$m_{ji}(x_i) = \sum_{x_j} \left\{ \phi_j(x_j)\phi_{i,j}(x_i, x_j) \prod_{k \in N(j)\setminus i} m_{kj}(x_j) \right\}.$$

- Once one node has all information, backtrack out to the leaves.

Homework: Third and Fourth UGM demos

For tomorrow, read/run the third and fourth demos:

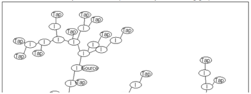
Tree UGM Demo

In the last demo we considered chain-structured data, one of the simplest types of dependency where we can take advantage of the graphical structure to allow efficient decoding/inference/sampling. In this demo, we consider the case of tree-structured graphical models. In particular, we consider the case where the edges in the graph can be arbitrary, as long as the graph doesn't contain any loops. For models with this type of dependency structure, we can still perform efficient decoding/inference/sampling by applying generalizations of the methods designed for chain-structured models.

Water Turbidity Problem

In some cities, it occasionally happens that snow melting off of the mountain causes an increase in the turbidity of the drinking water at some locations. The turbidity level is used as a surrogate for testing whether the water is safe to drink. For this reason, we may want to build a probabilistic model of the water turbidity at different locations in the water system. We will do this with a UGM, where we use a tree-structured model to capture the dependencies between connected elements of the water system.

We will assume that turbidity is measured on a scale of 1 to 4, where 1 represents 'very safe', and 4 represents 'very unsafe'. We will assume that the water system can be represented by the following graph:



Condition UGM Demo

In the previous three demos, we considered the *unconditional* decoding/inference/sampling tasks. That is, we assumed that we don't know the values of any of the random variables in the model. However, in many cases, we will want to consider what happens when we know the value of one or more of the random variables. That is, we have 'observations' and we want to do *conditional* decoding/inference /sampling.

For example, we might want to answer queries about the three previous demos like:

- Demo 1: If Mark and Cathy get the question wrong, what is the probability that Heather still gets the question right?
- Demo 2: What is the most likely path of a CS graduate's career, given that he is in academia 10 years after graduating? And what do samples of his career look like?
- Demo 3: What happens to the rest of the water system if the source node is in state 4? If we use a model with multiple sources and observe that one of the nodes is in state 4, which source is more likely to also be in an unsafe state?

Conditioning

UGMs are *closed* under conditioning. This means that if we condition on the values of some of the variables, the resulting distribution will still be a UGM. For example, consider the 4-node UGM with a chain-structured dependency 1-2-3-4, and the case where we want to condition on nodes 2 and 3. We obtain:

$$
\begin{aligned}
p(x_1, x_4 | x_2, x_3) &= \frac{p(x_1, x_2, x_3, x_4)}{p(x_2, x_3)} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \frac{1}{Z}\phi_1(x_1')\phi_2(x_2)\phi_3(x_3)\phi_4(x_4')\phi_1(x_1', x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4')} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\frac{1}{Z}\phi_2(x_2)\phi_3(x_3)\phi_2(x_2, x_3)\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1(x_1)\phi_4(x_4)\phi_1(x_1, x_2)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1'(x_1)\phi_4'(x_4)}{\sum_{x_1', x_4'} \phi_1'(x_1')\phi_4'(x_4')}
\end{aligned}
$$

Reviews/expands on material from today, introduces conditioning.

Discussion

- Exact decoding/inference/sampling is intractable in general.
- But it’s very efficient for graphs without loops.
- Tomorrow: ‘simple’ loops and conditional UGMs.