Graphical Models - Part II
Greg Mori - CMPT 419/726

Bishop PRML Ch. 8
Outline

Markov Random Fields

Inference
Outline

Markov Random Fields

Inference
Conditional Independence in Graphs

- Recall that for Bayesian Networks, conditional independence was a bit complicated
  - \textit{d-separation} with head-to-head links
- We would like to construct a graphical representation such that conditional independence is straight-forward path checking
Markov random fields (MRFs) contain one node per variable

Undirected graph over these nodes

Conditional independence will be given by simple separation, blockage by observing a node on a path

- e.g. in above graph, $A \perp \perp B | C$
With this simple check for conditional independence, **Markov blanket** is also simple

- Recall Markov blanket $MB$ of $x_i$ is set of nodes such that $x_i$ conditionally independent from rest of graph given $MB$

- Markov blanket is neighbours
Markov Random Fields

MRF Factorization

- Remember that graphical models define a factorization of the joint distribution.
- What should be the factorization so that we end up with the simple conditional independence check?
- For $x_i$ and $x_j$ not connected by an edge in graph:

  $$x_i \perp \perp x_j | x\backslash \{i,j\}$$

- So there should not be any factor $\psi(x_i, x_j)$ in the factorized form of the joint.
Clique

A **clique** in a graph is a subset of nodes such that there is a link between every pair of nodes in the subset.

A **maximal clique** is a clique for which one cannot add another node and have the set remain a clique.
MRF Joint Distribution

- Note that nodes in a clique cannot be made conditionally independent from each other
  - So defining factors \( \psi(\cdot) \) on nodes in a clique is “safe”
- The joint distribution for a Markov random field is:

\[
p(x_1, \ldots, x_K) = \frac{1}{Z} \prod_C \psi_C(x_C)
\]

where \( x_C \) is the set of nodes in clique \( C \), and the product runs over all maximal cliques
- Each \( \psi_C(x_C) \geq 0 \)
- \( Z \) is a normalization constant
The joint distribution for a Markov random field is:

\[
p(x_1, \ldots, x_4) = \frac{1}{Z} \prod_C \psi_C(x_C)
\]

\[
= \frac{1}{Z} \psi_{123}(x_1, x_2, x_3) \psi_{234}(x_2, x_3, x_4)
\]

Note that maximal cliques subsume smaller ones: \(\psi_{123}(x_1, x_2, x_3)\) could include \(\psi_{12}(x_1, x_2)\), though sometimes smaller cliques are explicitly used for clarity.
MRF Joint - Terminology

- The joint distribution for a Markov random field is:

\[ p(x_1, \ldots, x_K) = \frac{1}{Z} \prod_C \psi_C(x_C) \]

- Each \( \psi_C(x_C) \) is called a potential function

- \( Z \), the normalization constant, is called the partition function:

\[ Z = \sum_x \prod_C \psi_C(x_C) \]

- \( Z \) is very costly to compute, since it is a sum/integral over all possible states for all variables in \( x \)

- Don’t always need to evaluate it though, will cancel for computing conditional probabilities
Hammersley-Clifford

- The definition of the joint:

\[ p(x_1, \ldots, x_K) = \frac{1}{Z} \prod_C \psi_C(x_C) \]

- Note that we started with particular conditional independences
- We then formulated the factorization based on clique potentials
  - This formulation resulted in the right conditional independences
- The converse is true as well, any distribution with the conditional independences given by the undirected graph can be represented using a product of clique potentials
- This is the Hammersley-Clifford theorem
Often use exponential, which is non-negative, to define potential functions:

$$\psi_C(x_C) = \exp\{-E_C(x_C)\}$$

- Minus sign – by convention
- $E_C(x_C)$ is called an energy function
  - From physics, low energy = high probability
- This exponential representation is known as the Boltzmann distribution
Energy Functions - Intuition

- Joint distribution nicely rearranges as

\[ p(x_1, \ldots, x_K) = \frac{1}{Z} \prod_C \psi_C(x_C) \]

\[ = \frac{1}{Z} \exp\{- \sum_C E_C(x_C)\} \]

- Intuition about potential functions: \( \psi_C \) are describing good (low energy) sets of states for adjacent nodes

- An example of this is next
Consider the problem of trying to correct (denoise) an image that has been corrupted.

Assume image is binary.

Observed (noisy) pixel values $y_i \in \{-1, +1\}$

Unobserved true pixel values $x_i \in \{-1, +1\}$
Image Denoising - Graphical Model

- Cliques containing each true pixel value $x_i \in \{-1, +1\}$ and observed value $y_i \in \{-1, +1\}$
- Observed pixel value is usually the same as true pixel value
- Energy function $-\eta x_i y_i$, $\eta > 0$, lower energy (better) if $x_i = y_i$
- Cliques containing adjacent true pixel values $x_i, x_j$
- Nearby pixel values are usually the same
- Energy function $-\beta x_i x_j$, $\beta > 0$, lower energy (better) if $x_i = x_j$
Image Denoising - Graphical Model

- Cliques containing each true pixel value $x_i \in \{-1, +1\}$ and observed value $y_i \in \{-1, +1\}$
  - Observed pixel value is usually same as true pixel value
  - Energy function $-\eta x_i y_i$, $\eta > 0$, lower energy (better) if $x_i = y_i$
Image Denoising - Graphical Model

- Cliques containing each true pixel value $x_i \in \{-1, +1\}$ and observed value $y_i \in \{-1, +1\}$
  - Observed pixel value is usually same as true pixel value
  - Energy function $-\eta x_i y_i$, $\eta > 0$, lower energy (better) if $x_i = y_i$
- Cliques containing adjacent true pixel values $x_i, x_j$
  - Nearby pixel values are usually the same
  - Energy function $-\beta x_i x_j$, $\beta > 0$, lower energy (better) if $x_i = x_j$
Image Denoising - Graphical Model

- Complete energy function:
  \[ E(x, y) = -\beta \sum_{i,j} x_i x_j - \eta \sum_i x_i y_i \]

- Joint distribution:
  \[ p(x, y) = \frac{1}{Z} \exp\{-E(x, y)\} \]

- Or, as potential functions
  \[ \psi_n(x_i, x_j) = \exp(\beta x_i x_j), \]
  \[ \psi_p(x_i, y_i) = \exp(\eta x_i y_i) : \]
  \[ p(x, y) = \frac{1}{Z} \prod_{i,j} \psi_n(x_i, x_j) \prod_i \psi_p(x_i, y_i) \]
The denoising query is $\arg \max_x p(x|y)$

Two approaches:

- **Iterated conditional modes (ICM):** hill climbing in $x$, one variable $x_i$ at a time
  - Simple to compute, Markov blanket is just observation plus neighbouring pixels
- **Graph cuts:** formulate as max-flow/min-cut problem, exact inference (for this graph)
Consider a simple directed chain graph:

\[ p(x) = p(x_1)p(x_2|x_1)p(x_3|x_2) \ldots p(x_N|x_{N-1}) \]

Can convert to undirected graph
• Consider a simple directed chain graph:

\[ p(x) = p(x_1)p(x_2|x_1)p(x_3|x_2) \ldots p(x_N|x_{N-1}) \]

• Can convert to undirected graph

\[ p(x) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \ldots \psi_{N-1,N}(x_{N-1}, x_N) \]

where \( \psi_{1,2} = p(x_1)p(x_2|x_1) \), all other \( \psi_{k-1,k} = p(x_k|x_{k-1}) \), 
\( Z = 1 \)
The chain was straightforward because for each conditional \( p(x_i | pa_i) \), nodes \( x_i \cup pa_i \) were contained in one clique

- Hence we could define that clique potential to include that conditional

For a general undirected graph we can force this to occur by “marrying” the parents

- Add links between all parents in \( pa_i \)
- This process known as moralization, creating a moral graph
Strong Morals

- Start with directed graph on left
- Add undirected edges between all parents of each node
- Remove directionality from original edges
Constructing Potential Functions

- Initialize all potential functions to be 1
- With moral graph, for each $p(x_i|pa_i)$, there is at least one clique which contains all of $x_i \cup pa_i$
  - Multiply $p(x_i|pa_i)$ into potential function for one of these cliques
- $Z = 1$ again since:

\[
p(x) = \prod_C \psi_C(x_C) = \prod_i p(x_i|pa_i)
\]

which is already normalized
Equivalence Between Graph Types

- Note that the moralized undirected graph loses some of the conditional independence statements of the directed graph.
- Further, there are certain conditional independence assumptions which can be represented by directed graphs which cannot be represented by undirected graphs, and vice versa.
• Note that the moralized undirected graph loses some of the conditional independence statements of the directed graph.

• Further, there are certain conditional independence assumptions which can be represented by directed graphs which cannot be represented by undirected graphs, and vice versa.
Equivalence Between Graph Types

• Note that the moralized undirected graph loses some of the conditional independence statements of the directed graph.

• Further, there are certain conditional independence assumptions which can be represented by directed graphs which cannot be represented by undirected graphs, and vice versa.

• Directed graph: $A \perp B \mid \emptyset$, $A \not\perp B \mid C$, cannot be represented using undirected graph.
• Note that the moralized undirected graph loses some of the conditional independence statements of the directed graph

• Further, there are certain conditional independence assumptions which can be represented by directed graphs which cannot be represented by undirected graphs, and vice versa

• Directed graph: \( A \perp B | \emptyset \), \( A \perp B | C \), cannot be represented using undirected graph
• Note that the moralized undirected graph loses some of the conditional independence statements of the directed graph.

• Further, there are certain conditional independence assumptions which can be represented by directed graphs which cannot be represented by undirected graphs, and vice versa.

• Directed graph: $A \perp\!\!\!\!\perp B|\emptyset$, $A \perp\!\!\!\!\perp B|C$, cannot be represented using undirected graph.

• Undirected graph: $A \perp\!\!\!\!\perp B|\emptyset$, $A \perp\!\!\!\!\perp B|C \cup D$, $C \perp\!\!\!\!\perp D|A \cup B$ cannot be represented using directed graph.
Outline

Markov Random Fields

Inference
Inference

- **Inference** is the process of answering queries such as
  \[ p(x_n | x_e = e) \]
- We will focus on computing **marginal posterior distributions**
  over single variables \( x_n \) using
  \[ p(x_n | x_e = e) \propto p(x_n, x_e = e) \]
- The proportionality constant can be obtained by enforcing
  \[ \sum_{x_n} p(x_n | x_e = e) = 1 \]
Inference on a Chain

- Consider a simple undirected chain
- For inference, we want to compute \( p(x_n, x_e = e) \)
- First, we’ll show how to compute \( p(x_n) \)
  - \( p(x_n, x_e = e) \) will be a simple modification of this
Inference on a Chain

- The naive method of computing the marginal $p(x_n)$ is to write down the factored form of the joint, and marginalize (sum out) all other variables:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_n-1} \sum_{x_{n+1}} \cdots \sum_{x_N} p(x)$$

$$= \sum_{x_1} \cdots \sum_{x_n-1} \sum_{x_{n+1}} \cdots \sum_{x_N} \frac{1}{Z} \prod_C \psi_C(x_C)$$

- This would be slow – $O(K^N)$ work if each variable could take $K$ values
Inference on a Chain

- However, due to the factorization terms in this summation can be rearranged nicely.
- This will lead to efficient algorithms.
Simple Algebra

• This efficiency comes from a very simple distributivity

\[ ab + ac = a(b + c) \]

• Or more complicated version

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} a_i b_j = a_1 b_1 + a_1 b_2 + \ldots + a_n b_n
\]

\[
= (a_1 + \ldots + a_n)(b_1 + \ldots + b_n)
\]

• Much faster to do right hand side \((2(n - 1) \text{ additions, } 1 \text{ multiplication})\) than left hand side \((n^2 \text{ multiplications, } n^2 - 1 \text{ additions})\)
A Simple Chain

First consider a chain with 3 nodes, and computing \( p(x_3) \):

\[
p(x_3) = \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3)
\]

\[
= \sum_{x_2} \psi_{23}(x_2, x_3) \left[ \sum_{x_1} \psi_{12}(x_1, x_2) \right]
\]
Performing the sums

\[ p(x_3) = \sum_{x_2} \psi_{23}(x_2, x_3) \left[ \sum_{x_1} \psi_{12}(x_1, x_2) \right] \]

- For example, if \( x_i \) are binary:

\[
\psi_{12}(x_1, x_2) = x_1 \begin{bmatrix} a & b \\ c & d \end{bmatrix}_{x_2} \quad \psi_{23}(x_2, x_3) = x_2 \begin{bmatrix} s & t \\ u & v \end{bmatrix}_{x_3}
\]
Performing the sums

\[ p(x_3) = \sum_{x_2} \psi_{23}(x_2, x_3) \left[ \sum_{x_1} \psi_{12}(x_1, x_2) \right] \]

- For example, if \( x_i \) are binary:

\[ \psi_{12}(x_1, x_2) = x_1 \begin{bmatrix} a & b \\ c & d \end{bmatrix} \]
\[ \psi_{23}(x_2, x_3) = x_2 \begin{bmatrix} s & t \\ u & v \end{bmatrix} \]

\[ \sum_{x_1} \psi_{12}(x_1, x_2) = \begin{bmatrix} a + c & b + d \end{bmatrix} \equiv \mu_{12}(x_2) \]
Performing the sums

\[ p(x_3) = \sum_{x_2} \psi_{23}(x_2, x_3) \left[ \sum_{x_1} \psi_{12}(x_1, x_2) \right] \]

- For example, if \( x_i \) are binary:

\[ \psi_{12}(x_1, x_2) = x_1 \begin{bmatrix} a & b \\ c & d \end{bmatrix} \]

\[ \psi_{23}(x_2, x_3) = x_2 \begin{bmatrix} s & t \\ u & v \end{bmatrix} \]

\[ \sum_{x_1} \psi_{12}(x_1, x_2) = \begin{bmatrix} a + c & b + d \end{bmatrix} \equiv \mu_{12}(x_2) \]

\[ \psi_{23}(x_2, x_3) \times \mu_{12}(x_2) = x_2 \begin{bmatrix} s(a + c) & t(a + c) \\ u(b + d) & v(b + d) \end{bmatrix} \]
Performing the sums

\[ p(x_3) = \sum_{x_2} \psi_{23}(x_2, x_3) \left[ \sum_{x_1} \psi_{12}(x_1, x_2) \right] \]

- For example, if \( x_i \) are binary:

\[ \psi_{12}(x_1, x_2) = x_1 \begin{bmatrix} a & b \\ c & d \end{bmatrix}_{x_2} \]

\[ \psi_{23}(x_2, x_3) = x_2 \begin{bmatrix} s & t \\ u & v \end{bmatrix}_{x_3} \]

\[ \sum_{x_1} \psi_{12}(x_1, x_2) = \begin{bmatrix} a + c & b + d \end{bmatrix}_{x_2} \equiv \mu_{12}(x_2) \]

\[ \psi_{23}(x_2, x_3) \times \mu_{12}(x_2) = x_2 \begin{bmatrix} s(a + c) & t(a + c) \\ u(b + d) & v(b + d) \end{bmatrix}_{x_3} \]

\[ p(x_3) = \begin{bmatrix} s(a + c) + u(b + d) & t(a + c) + v(b + d) \end{bmatrix} \]
Complexity of Inference

- There were two types of operations
  - Summation
    \[ \sum_{x_1} \psi_{12}(x_1, x_2) \]
    \( K \times K \) numbers in \( \psi_{12} \), takes \( O(K^2) \) time
  - Multiplication
    \[ \psi_{23}(x_2, x_3) \times \mu_{12}(x_2) \]
    Again \( O(K^2) \) work
- For a chain of length \( N \), we repeat these operations \( N - 1 \) times each
  - \( O(NK^2) \) work, versus \( O(K^N) \) for naive evaluation
More complicated chain

- Now consider a 5 node chain, again asking for $p(x_3)$

$$p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5)$$

- Each of these factors resembles the previous, and can be computed efficiently
- Again $O(NK^2)$ work
More complicated chain

- Now consider a 5 node chain, again asking for $p(x_3)$

\[
p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5)
\]

\[
= \sum_{x_2} \sum_{x_1} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \sum_{x_4} \sum_{x_5} \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5)
\]
More complicated chain

- Now consider a 5 node chain, again asking for $p(x_3)$

\[
p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5)
\]

\[
= \sum_{x_2} \sum_{x_1} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \sum_{x_4} \sum_{x_5} \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5)
\]

\[
= \left[ \sum_{x_2} \sum_{x_1} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \right] \left[ \sum_{x_4} \sum_{x_5} \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5) \right]
\]
More complicated chain

• Now consider a 5 node chain, again asking for $p(x_3)$

\[
p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5)
\]

\[
= \sum_{x_2} \sum_{x_1} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \sum_{x_4} \sum_{x_5} \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5)
\]

\[
= \left[ \sum_{x_2} \sum_{x_1} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \right] \left[ \sum_{x_4} \sum_{x_5} \psi_{34}(x_3, x_4) \psi_{45}(x_4, x_5) \right]
\]

• Each of these factors resembles the previous, and can be computed efficiently
  • Again $O(NK^2)$ work
The factors can be thought of as messages being passed between nodes in the graph.

\[ \mu_{12}(x_2) \equiv \sum_{x_1} \psi_{12}(x_1, x_2) \]

is a message passed from node \(x_1\) to node \(x_2\) containing all information in node \(x_1\).

In general,

\[ \mu_{k-1,k}(x_k) = \sum_{x_{k-1}} \psi_{k-1,k}(x_{k-1}, x_k) \mu_{k-2,k-1}(x_{k-1}) \]

Possible to do so because of conditional independence.
Computing All Marginals

- Computing one marginal $p(x_n)$ takes $O(NK^2)$ time
- Naively running same algorithms for all nodes in a chain would take $O(N^2K^2)$ time
- But this isn’t necessary, same messages can be reused at all nodes in the chain
- Pass all messages from one end of the chain to the other, pass all messages in the other direction too
- Can compute marginal at any node by multiplying the two messages delivered to the node
  - $2(N - 1)K^2$ work, twice as much as for just one node
Including Evidence

- If a node $x_{k-1} = e$ is observed, simply clamp to observed value rather than summing:

$$
\mu_{k-1,k}(x_k) = \sum_{x_{k-1}} \psi_{k-1,k}(x_{k-1}, x_k) \mu_{k-2,k-1}(x_{k-1})
$$

becomes

$$
\mu_{k-1,k}(x_k) = \psi_{k-1,k}(x_{k-1} = e, x_k) \mu_{k-2,k-1}(x_{k-1} = e)
$$
The algorithm for a tree-structured graph is very similar to that for chains

Formulation in PRML uses factor graphs, we’ll just give the intuition here

Consider calculating the marginal $p(x_n)$ for the center node of the graph at right

Treat $x_n$ as root of tree, pass messages from leaf nodes up to root
Markov Random Fields

Trees

- Message passing similar to that in chains, but possibly multiple messages reaching a node
- With multiple messages, multiply them together
- As before, sum out variables

\[
\mu_{k-1,k}(x_k) = \sum_{x_{k-1}} \psi_{k-1,k}(x_{k-1}, x_k) \mu_{k-2,k-1}(x_{k-1})
\]

- Known as sum-product algorithm
- Complexity still \(O(NK^2)\)
Most Likely Configuration

- A similar algorithm exists for finding

\[
\arg \max_{x_1, \ldots, x_N} p(x_1, \ldots, x_N)
\]

- Replace summation operations with maximize operations
- Maximum of products at each node
- Known as max-sum, since often take log-probability to avoid underflow errors
General Graphs

- Junction tree algorithm is an exact inference method for arbitrary graphs
  - A particular tree structure defined over cliques of variables
  - Inference ends up being exponential in maximum clique size
  - Therefore slow in many cases
- Approximate inference techniques
  - Loopy belief propagation: run message passing scheme (sum-product) for a while
    - Sometimes works
    - Not guaranteed to converge
  - Variational methods: approximate desired distribution using analytically simple forms, find parameters to make these forms similar to actual desired distribution (Ch. 10, we won’t cover)
  - Sampling methods: represent desired distribution with a set of samples, as more samples are used, obtain more accurate representation (Ch. 11, we will cover)
Conclusion

- Readings: Ch. 8
- Graphical models depict conditional independence assumptions
- Directed graphs (Bayesian networks)
  - Factorization of joint distribution as conditional on node given parents
- Undirected graphs (Markov random fields)
  - Factorization of joint distribution as clique potential functions
- Inference algorithm sum-product, based on local message passing
  - Works for tree-structured graphs
  - Non-tree-structured graphs, either slow exact or approximate inference