Loopy belief propagation

CS B553
Spring 2013

Announcements

• A2 due tonight
  — Reminder of late policy: 10% off for up to 48 hours late
• A3 coming soon!

Constructing a clique tree

• One way to find a clique tree is to choose a variable elimination ordering and “run” VE
• Another approach is to use a graph construction
  — If necessary, moralize to produce an undirected graph G.
  — Triangulate G to produce a chordal graph H. (Would like one with minimum clique size, but this is NP hard.)
  — Find maximal cliques in H. (Not NP hard for chordal graphs.)
  — Construct graph with nodes corresponding to max cliques in H, edges weighted according to degree of overlap. i.e. edge between C1 and C2 has weight |C1 ∩ C2|
  — Find a max spanning tree on this graph to yield a clique tree.

Finding a clique tree

(Clique trees for grid graphs)

• The tree width of a graph G is equal to m-1,
  — Where m is the size of the largest clique in the triangulated (chordal) version of G
• The worst-case running time of exact inference on a Markov or Bayes network is exponential in the tree width of the moralized graph.
  — E.g., the tree width of an n x n grid graph is n, so inference takes O(s^n) time, where s is # of possible values of each random variable

Clique trees for grid graphs

Tree width
Making inference tractable

• Three strategies for efficient inference in practice:
  1. Use a graphical model with low tree width.
  2. Make some additional assumptions about the problem.
     (Efficient algorithms exist for some special cases, e.g. when the clique potentials have a specific form.)

Making inference tractable

• In practice, making inference tractable is a key challenge in applying graphical models to applications
  • Typically, the options are:
    - Exact inference with arbitrary potentials on a graphical model, but with a simplified structure
    - Exact inference on a graphical model with arbitrary structure, but restricted potentials
    - Graphical model with arbitrary structure and arbitrary potentials, but settle for approximate inference

Approximate inference

• So far, we’ve covered exact inference algorithms
  - i.e. algorithms that can compute marginal probability distributions exactly, with no error
  - They are fast for some graphical models, like trees, but are prohibitively expensive for most applications
  • But is exact inference worth it?
    - Our models aren’t perfect anyway…
  • Much more efficient approximate inference algorithms exist
    - If we’re willing to settle for inexact answers

Consider a simple cycle:

• A clique tree for this cycle is:
  - Recall that a clique tree is a special cluster graph (each node is a clique of the original graph, edges connect subset of nodes sharing a common variable)
  • Here’s another cluster graph for the above cycle:
    - It’s not a clique tree
    - It’s a loopy graph

Back to sum-product…

• Recall that sum-product involves sending messages to neighbors,
  \[ \delta_{i\rightarrow j}(S_{k, j}) = \sum_{C_i \cap C_j} \psi(C_i) \prod_{k \in \mathcal{N}(i) \setminus \{j\}} \delta_{k\rightarrow i}(S_{k, i}) \]
  • We derived this for use on a clique tree, but the definition of messages (above) works on any graph!
    - i.e., there’s nothing that prevents us from running Sum-Product Belief Propagation on a loopy cluster graph
    - There’s just no theoretical justification for doing this…

Sum product

\[ \delta_{i\rightarrow j}(S_{k, j}) = \sum_{C_i \cap C_j} \psi(C_i) \prod_{k \in \mathcal{N}(i) \setminus \{j\}} \delta_{k\rightarrow i}(S_{k, i}) \]

BP on a clique tree
• Messages start at root, then propagate up to leaves, then propagate down to leaves

BP on loopy cluster graph
• Nodes send messages to their neighbors, based on their clique potential and messages from neighbors
Loopy Belief Propagation

- Construct a cluster graph from the Markov network
- Each node sends messages to its neighbor,
  \[ \delta_{\rightarrow j}(S_{ik}) = \sum_{C_j} \psi_i(C_i) \prod_{k \in N(i) \setminus \{j\}} \delta_{\rightarrow k}(S_{ik}) \]
- Problems with this?
- How to begin?
  - Initialize by having every node send a “fake” initial message, consisting of just a uniform distribution
- How to end?
  - Keep running iterations of BP until convergence
  - Unfortunately, convergence might not happen...

Historical note: Turbo codes

- Loopy Belief Propagation (LBP) has been known for ~25 years, but people assumed it was not useful
- In the 1990s, a seemingly unrelated discovery sparked renewed interest in LBP
  - Paper on a new error-correcting transmission algorithm, for noisy links (e.g. wireless networks)
  - Breakthrough in communications technology; modern wireless phones still use this (or a related) technique

Loopy BP properties

- Recall that clique trees exhibited the running intersection property:
  - If variable X is part of node C and node D, then every node along the path from C to D also contains X
- Loopy cluster graphs won’t satisfy this condition
- Instead, we’ll require the following weaker property:
  - If variable X is part of node C and node D, then there exists exactly one path between C and D such that the scope of the messages along the path include X
  - Intuitively, helps to prevent endless propagation cycles

Turbo codes

- Berrou (1993)
- Solve for U1—U4, given Y1—Y8
- Hack: First solve for Y1, Y3, Y5, Y7.
- Hold these constant, then solve for Y2, Y4, Y6, Y8.
- Then repeat, iteratively; i.e. “Turbocharge” estimates using iterative feedback loop
- Unbeknownst to them, they had (re-)discovered Loopy Belief Propagation!

Noisy link model

- Suppose we want to send bits U1, U2, U3, U4 across a noisy communications link
  - Need some redundancy to detect and correct errors
  - So, we encode these bits as some new bits X1—X8
  - The observer receives some bits Y1—Y8, which may be a corrupted copy of X1—X8
- From a probabilistic inference standpoint, receiver wants to compute distribution over X given Y

Loopy BP properties

- Loopy BP can be much faster than exact inference
  - Running time?
- Loopy BP is not guaranteed to converge to a solution
  - Even if it does converge, it’s not guaranteed to converge to the correct solution
    - However, in practice it seems to converge to a reasonably good solution for a variety of Markov net problems
- Rather little is known about its theoretical properties
  - Including when it will work well and when it won’t
    - We’ll see (later) one explanation for why it works
Loopy BP on grid graphs

- In the special case of grid graphs, we can run BP on the Markov net directly, instead of the cluster graph.
- This is possible because we can create a cluster graph with a very simple structure.
- The computation is exactly the same, but it’s more convenient to understand and implement this way.

Belief propagation

- Messages are passed between the variables.
  - Message from node $i$ to $j$ at iteration $t$ is:
    \[
    m_{i ightarrow j}^{(t)}(X_j) = \sum_{X_i} \phi(X_i, X_j) \prod_{k \in N(X_i) \setminus \{X_j\}} m_{k ightarrow i}^{(t-1)}(X_i)
    \]
  - Intuitively (and anthropomorphically) a message from me to you says, for each of your states, “If you choose state $X_j$, here’s how happy my neighbors and I would be.”