Bookmarks

Introduction

Part 1: Probability and Inference

Part 2: Inference in Graphical Models

Week 5: Introduction to Part 2 on Inference in Graphical Models

Week 5: Efficiency in Computer Programs
Exercises due Oct 20, 2016 at 02:30 IST

Week 5: Graphical Models
Exercises due Oct 20, 2016 at 02:30 IST

Week 5: Homework 4
Homework due Oct 21, 2016 at 02:30 IST

Week 6: Inference in Graphical Models - Marginalization

Part 2: Inference in Graphical Models > Week 6: Inference in Graphical Models - Marginalization > Exercise: Speeding Up Sum-Product

Exercise: Speeding Up Sum-Product

Bookmark this page

Exercise: Speeding Up Sum-Product

11/11 points (graded)

As we saw in the earlier exercise, the sum-product algorithm naively implemented can take time that is more than linear in the number of nodes n. We can see this from the worst-case star graph:

Exercises due Oct 27, 2016 at 02:30 IST

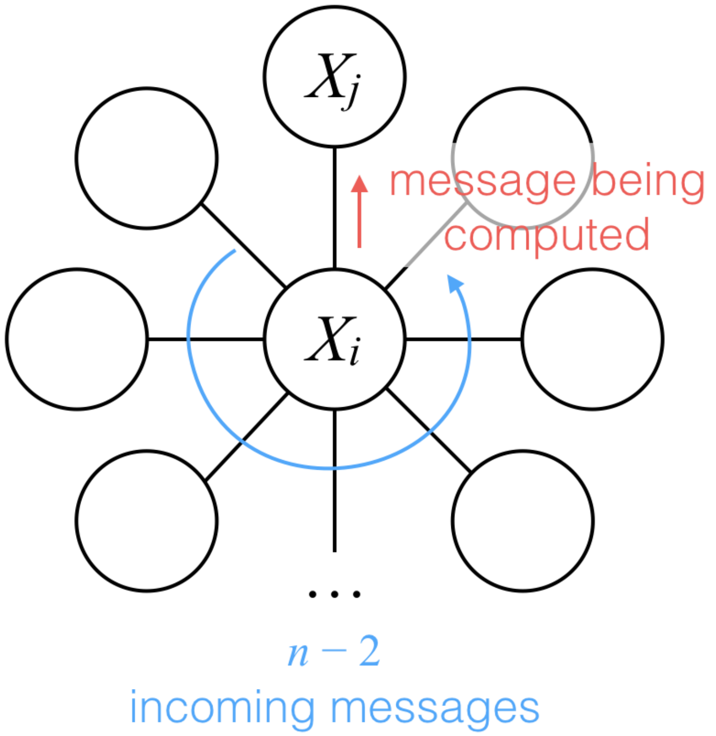
Week 6: Special Case - Marginalization in Hidden Markov Models
Exercises due Oct 27, 2016 at 02:30 IST

Week 6: Homework 5
Homework due Oct 27, 2016 at 02:30 IST

Weeks 6 and 7: Mini-project on Robot Localization
Mini-projects due Nov 03, 2016 at 02:30 IST

Week 7: Inference with Graphical Models - Most Probable Configuration
Exercises due Nov 03, 2016 at 02:30 IST

Week 7: Special Case - MAP Estimation in Hidden Markov Models



In particular with node i referring to the center of the star graph, for every possible node j that is not i (there are $\mathcal{O}(n)$ choices for j), the computation of $m_{i\to j}$ requires taking the product of $\mathcal{O}(n)$ terms, so we get hit by a total running time that at least scales with n^2.

It turns out that with clever implementation, we can cut this running time down to linear in n. We walk through a way of doing this, which will treat the two passes of the sum-product slightly differently in terms of calculation. Note that this way of speeding up the calculation will require that the message tables always have strictly positive entries. There are ways around this assumption that still keeps the computation linear in n but to keep the exposition here simple we'll stick to strictly positive message table entries.

In what follows, let d_i denote the number of neighbors that node i has; in graph theory, d_i is called the *degree* of node i. As before, assume that every X_i takes on one of k possible values (note that even if the different X_i's took on a different number of values, we can take k to be the maximum alphabet size of any of the X_i's to get an upper bound).

Sum-Product: Message Passing from the Leaves to the Root

After choosing an arbitrary node to be the root of the tree, then we know what the leaves are, and every node (except the root node) has one unique parent, where if we look at any leaf node, take its parent node, then the parent node of that parent node, and so forth, eventually we will always reach the root node.

Let $\pi(i)$ denote the parent of non-root node i. Then when passing messages from leaves to the root node, the messages are of the form:

$$m_{i\to\pi(i)}(x_{\pi(i)}) = \sum_{x_i}\left[\psi_{i,\pi(i)}(x_i, x_{\pi(i)})\underbrace{\phi_i(x_i)\prod_{\ell\in\mathcal{N}(i)\text{ such that }\ell\neq\pi(i)} m_{\ell\to i}(x_i)}_{\text{define this to be }\xi_i(x_i)}\right],$$

The reason for defining a table ξ_i (which has one entry for every possible value in the alphabet of X_i) is that we will use ξ_i during message passing from the root node back to the leaves that will eliminate redundant calculation.

- When passing messages from leaves to the root, for a non-root node, exactly how many messages are sent from it?

1 ✓

For non-root node i, let's break the computation of $m_{i\to\pi(i)}$ into two steps. First we compute the table ξ_i:

$$\xi_i(x_i) = \phi_i(x_i) \prod_{\ell \in \mathcal{N}(i) \text{ such that } \ell \neq \pi(i)} m_{\ell \to i}(x_i).$$

- For a specific non-root node i, what is the running time of computing the table ξ_i (again, the messages that it depends on have already been computed because of the ordering in which the sum-product algorithm computes messages)?

  Choose the answer with **smallest** big O bound in terms of d_i and k (unless one of these doesn't matter).

  ○ $\mathcal{O}(d_i)$

  ● $\mathcal{O}(d_i k)$ ✓

  ○ $\mathcal{O}(d_i k^2)$

  ○ $\mathcal{O}(d_i k^3)$

After computing the table ξ_i, we compute

$$m_{i\to\pi(i)}(x_{\pi(i)}) = \sum_{x_i} \psi_{i,\pi(i)}(x_i, x_{\pi(i)})\xi_i(x_i).$$

- For a specific non-root node i, what is the running time of computing the table $m_{i\to\pi(i)}$ given that we have already computed ξ_i?

  Choose the answer with **smallest** big O bound in terms of d_i and k (unless one of these doesn't matter).

○ $\mathcal{O}(k)$

● $\mathcal{O}(k^2)$ ✓

○ $\mathcal{O}(k^3)$

○ $\mathcal{O}(d_i)$

○ $\mathcal{O}(d_i k)$

○ $\mathcal{O}(d_i k^2)$

○ $\mathcal{O}(d_i k^3)$

For a tree with n nodes, note that d_i is the number of edges that node i participates in, and so $\sum_{i=1}^{n} d_i$ is the sum of the number of edges that every node participates in, where we count each edge multiple times since multiple nodes can participate in the same edge.

- Determine a simple expression for $\sum_{i=1}^{n} d_i$ that is in terms of n.

  In this part, please provide your answer as a mathematical formula (and not as Python code). Use `^` for exponentiation, e.g., `x^2` denotes x^2. Explicitly include multiplication using `*`, e.g. `x*y` is xy.

  2*(n-1) ✓

  $2 \cdot (n - 1)$

Using your answers to the previous parts, the total number of operations for computing messages from leaves to the root can be decomposed into the number of operations for computing all the ξ_i's plus the number of operations for computing every message $m_{i \to \pi(i)}$ where ξ_i is already computed.

- What is the number of operations for computing all the ξ_i's? Note that a summation over all nodes i that is not the root node can be upper-bounded by a summation over all nodes.

  Choose the answer with **smallest** big O bound in terms of k and n (unless one of these doesn't matter).

  ○ $\mathcal{O}(k)$

  ○ $\mathcal{O}(k^2)$

  ○ $\mathcal{O}(k^3)$

● $\mathcal{O}(nk)$ ✓

○ $\mathcal{O}(nk^2)$

○ $\mathcal{O}(nk^3)$

- What is the number of operations for computing all the messages of the form $m_{i\to\pi(i)}$ (where when computing the message from i to $\pi(i)$, we have already computed ξ_i)?

  Choose the answer with **smallest** big O bound in terms of k and n (unless one of these doesn't matter).

○ $\mathcal{O}(k)$

○ $\mathcal{O}(k^2)$

○ $\mathcal{O}(k^3)$

○ $\mathcal{O}(nk)$

● $\mathcal{O}(nk^2)$ ✓

○ $\mathcal{O}(nk^3)$

- What is the total number of operations for passing messages from leaves to the root?

  Choose the answer with **smallest** big O bound in terms of k and n (unless one of these doesn't matter).

  ○ $\mathcal{O}(k)$

  ○ $\mathcal{O}(k^2)$

  ○ $\mathcal{O}(k^3)$

  ○ $\mathcal{O}(nk)$

  ● $\mathcal{O}(nk^2)$ ✓

  ○ $\mathcal{O}(nk^3)$

Passing messages the other direction is where we have to be careful to avoid the disaster encountered with the star-shape graph. We can avoid this disaster by keeping track of (possibly unnormalized) marginal distributions as we pass messages from the root back to the leaves.

Sum-Product: Computing the Marginal Distribution at the Root

Note that as we saw in the video/course notes on the sum-product algorithm, once we compute all the messages from leaves to the root, then we can compute the marginal distribution for the root node r.

Let's denote the unnormalized marginal distribution for X_r where r is the root node as

$$\tilde{p}_{X_r}(x_r) \triangleq \phi_r(x_r) \prod_{j \in \mathcal{N}(r)} m_{j \to r}(x_r).$$

- What is the number of operations for computing unnormalized marginal $\tilde{p}_{X_r}$?

  Choose the answer with **smallest** big O bound that is in terms of d_r and k (unless one of these does not matter).

  ○ $\mathcal{O}(d_r)$

  ● $\mathcal{O}(d_r k)$ ✓

  ○ $\mathcal{O}(d_r k^2)$

  ○ $\mathcal{O}(d_r k^3)$

Of course, computing the normalized marginal p_{X_r} is straightforward as we just divide the entries of unnormalized marginal $\tilde{p}_{X_r}$ by the sum of its entries, which costs $\mathcal{O}(k)$ operations.

Sum-Product: Message Passing from the Root to the Leaves

The basic idea is that we write the message passing equation now in terms of the unnormalized marginal of whichever node we are passing the message from.

We start by passing messages from the root node to its neighboring nodes:

$$\begin{aligned} m_{r\to j}(x_j) &= \sum_{x_r}\left[\phi_r(x_r)\psi_{r,j}(x_r,x_j)\prod_{\ell\in\mathcal{N}(r)\text{ such that }\ell\neq j} m_{\ell\to r}(x_r)\right] \\ &= \sum_{x_r}\frac{\psi_{r,j}(x_r,x_j)\tilde{p}_{X_r}(x_r)}{m_{j\to r}(x_r)}, \end{aligned}$$

where here is where we make the assumption that all the message table entries are strictly positive, so as to not run into a division by 0 issue. Importantly, we have already computed all the tables inside the summation!

- What is the total number of operations for computing table $m_{r\to j}$ given that we already have tables $\psi_{r,j}, \tilde{p}_{X_r}$, and $m_{j\to r}$ available?

  Choose the answer with **smallest** big O bound that is in terms of d_r and k (unless one of these does not matter).

  ○ $\mathcal{O}(k)$

  ● $\mathcal{O}(k^2)$ ✓

  ○ $\mathcal{O}(k^3)$

○ $\mathcal{O}(d_r)$

○ $\mathcal{O}(d_r k)$

○ $\mathcal{O}(d_r k^2)$

○ $\mathcal{O}(d_r k^3)$

Right after we compute $m_{r\to j}$ for each neighbor j of root node r, we can compute the unnormalized marginals for each neighbor node j:

$$\tilde{p}_{X_j}(x_j) \triangleq \xi_j(x_j) m_{\pi(j)\to j}(x_j).$$

This is where the ξ_i's that we computed earlier come back into play!

- What is the number of operations for computing unnormalized marginal $\tilde{p}_{X_j}$ for a specific node j?

  Choose the answer with **smallest** big O bound that is in terms of d_j and k (unless one of these does not matter).

  ● $\mathcal{O}(k)$ ✓

  ○ $\mathcal{O}(k^2)$

○ $\mathcal{O}(k^3)$

○ $\mathcal{O}(d_j)$

○ $\mathcal{O}(d_j k)$

○ $\mathcal{O}(d_j k^2)$

○ $\mathcal{O}(d_j k^3)$

At this point, we can actually just repeat the same idea since in general for any node i and its neighbor j,

$$m_{i\to j}(x_j) = \sum_{x_i} \frac{\psi_{i,j}(x_i, x_j)\tilde{p}_{X_i}(x_i)}{m_{j\to i}(x_i)}.$$

In particular, we keep passing messages, and after passing each message, we compute an unnormalized marginal (and also normalizing so that we get all the marginal distributions).

- Using this strategy outlined above, how many operations does computing all the messages from the root back to the leaves and computing all the marginals take?

○ $\mathcal{O}(k)$

○ $\mathcal{O}(k^2)$

○ $\mathcal{O}(k^3)$

○ $\mathcal{O}(nk)$

● $\mathcal{O}(nk^2)$ ✓

○ $\mathcal{O}(nk^3)$

Overall, the running time is now linear in n with careful storage of some extra tables and reordering some of the computations. In practice, how we presented the sum-product algorithm previously is cleaner to code up and is typically fast enough as we aren't dealing with bad graph cases such as the star graph.

For you to think about: Can you see how to handle the case when message table entries can be 0 but still using the above strategy for speeding up the computation?

Submit You have used 1 of 5 attempts

✓ Correct (11/11 points)

Discussion

Topic: Inference in Graphical Models - Marginalization / Exercise: Speeding Up Sum-Product

Hide Discussion

Add a Post

mrBB

about 3 hours ago

A node can have a maximum of $n-1$ edges: to all nodes in the graph except to itself. Of these $n-1$ edges, one is outgoing and the rest, the other $n-2$, are incoming.

Add a comment

Showing all responses

Add a response:

Preview

Collapse discussion

So much effort

discussion posted a day ago by **yasser-5**

Guys, I hardly got 10 out of 11 in this exercise and the previous one after painful thinking and focusing for more than 4 hours I've a headache right now!

I badly need any auxiliary resources, any YouTube playlist,... in order to spare some time and effort. I spend tooooo much time than necessary trying to trace the material.

Thanks

This post is visible to everyone.

Add A Response

1 response

borowis

about 15 hours ago

Spoiler: unfortunately, no links.

It's hard to say how much time is necessary. I sometimes feel myself completely stupid. The most important point is to think, think, think, and not to give up. It should come.

Collapse discussion

about 8 hours ago

Only George can but in 6.00.1x this sort of thing normally means that there was a hidden character pasted into the box from an editor. Damned hard to detect although they show up in the edx machinery somehow.

Add a comment

Showing all responses

Add a response:

Preview

Submit

Collapse discussion

Problem with part 9

question posted about 9 hours ago by **Teppakorn**

Sum-Product: Message Passing from the Root to the Leaves

$$m_{r\to j}(x_j) = \sum_{x_r}\left[\phi_r(x_r)\psi_{r,j}(x_r,x_j)\prod_{\ell\in\mathcal{N}(r)\text{ such that }\ell\neq j} m_{\ell\to r}(x_r)\right]$$
$$= \sum_{x_r}\frac{\psi_{r,j}(x_r,x_j)\tilde{p}_{X_r}(x_r)}{m_{j\to r}(x_r)},$$

Why not take summation of a denominator first before taking all of xr.

$$= \sum_{x_r}\frac{\psi_{r,j}(x_r,x_j)\tilde{p}_{X_r}(x_r)}{\sum_{x_r} m_{j\to r}(x_r)},$$

This post is visible to everyone.

Add A Response

1 response

Mark_B2 **Community TA**

about 8 hours ago

Collapse discussion

Minor typos

discussion posted 8 days ago by **Grimeson**

"maximize alphabet size" should be "maximum alphabet size."

edit: "computing all every" -> "computing every"

edit2: "with carefully storage" -> "with careful storage"

This post is visible to everyone.

Add A Response

1 response

georgehc **Staff**

8 days ago

Thanks for pointing this out! Fixed -George Chen (instructor)

Some more typos:

> Right after we compute $m_{r\to j}$ for each neighbor j of root node
> i (r)

Collapse discussion

RADUGROSU **Community TA**

4 days ago

You omitted to specify what the perceived ambiguity was.

I found the 'it' at the end of the sentence to be ambiguous. I feel the sentence can be better worded as follows:

When passing messages from leaves to the root, exactly how many messages are sent from a non-root node?

posted 4 days ago by **kejriwall**

I would have probably used a phrasing similar to yours, but I have to say I cannot see the source of ambiguity. The only other singular noun around, *root*, was farther away from the *it* than the actual antecedent, *non-root node*. Moreover, the semantic only strengthens this parsing.

posted 4 days ago by **RADUGROSU** **Community TA**

It is bad english because it creates potential ambiguity. Explicit is better when so much time is spent guessing and searching for examples where anyone else does something similar. I am guessing that the intended meaning was:

When passing messages from leaves to the root, exactly how many messages are

− **Collapse discussion**

For the Determine a simple expression..

discussion posted 8 days ago by **Ryan_mS**

Should accept alternate form of answer.

This post is visible to everyone.

Add a response:

0 responses

Preview

Submit

− Collapse discussion

1

© All Rights Reserved

© 2016 edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.

POWERED BY
OPENedX

Download on the
App Store

ANDROID APP ON
Google play