Belief Propagation on MRFs

David Doria

February 23, 2011

1 Introduction

Belief propagation (BP) is a technique which can be used to find the optimal labeling of a graphical model. BP can be defined and explained using different terminology and notation for Bayesian networks, factor graphs, and Markov random fields (MRFs). We have chosen to exclusively use the notation of MRFs throughout this discussion. This choice greatly simplifies the notation, as the energy functions are a function only of the labels of a pair of neighboring nodes, whereas in the more general framework of factor graphs, these functions are a function of the node labels as well as the node ids. In this document we explain the "sum-product message passing" algorithm for performing this optimization.

2 Warning

On graphicals model with a tree structure, belief propagation is an exact optimization method. That is, it is guaranteed to produce the correct result and it is guaranteed to converge. However, MRFs do not exhibit this tree structure - there are many "loops" in the graph of an MRF. With only a slight modification (described in Section 7), the BP algorithm has been shown to "usually work" on loopy graphs. When BP is applied to a loopy graphical model, it is known as Loopy Belief Propagation, or Loopy BP.

3 Notation

3.1 Neighbors

We will often need to refer to the neighbors of a node. We define the function $N(x)$ to mean all neighbors of node x.

$$N(x) = \text{All neighbors of node x} \tag{1}$$

We will also need to refer to all of the neighbors of a node except for a particular node. We use the set theoretic notation $N(x)\backslash y$ to mean all neighbors of node x except for y.

$$N(x)\backslash y = \text{All neighbors of node x except for y} \tag{2}$$

3.2 Labels

3.2.1 Refering to a node's label

We will use the notation l_n to refer to the label of node n.

3.2.2 Possible label set

The set of possible labels is denoted L. The size of the label set $|L|$ is the number of possible labels. The i^{th} label is denoted $L(i)$.

4 Terminology

The idea of a message passing algorithm is for each node in a graphical model to "tell" its neighbors what it "believes" about their labeling. As a simple example, in a binary labeling problem, node A might "say" to node B "Hey node B, I'm pretty sure that you should be label 0". Of course, this notion of "pretty sure" is more formally defined as a value similar to a likelihood.

A "message" is "passed" from a node i to one of its neighbors node j indicating the the amount that node i "believes" that node j should be a particular label. One of these messages must be sent for each label in the label set L. That is, if node j could take one of $|L|$ labels, there are $|L|$ messages to be passed from node i to node j.

5 Energy functions

5.1 Unary

The unary energy function defines the energy (cost) for any node to take the label $L(x)$.

$$U(L(x)) = \text{Energy of a node taking the label } L(x) \tag{3}$$

5.2 Binary

The binary energy function defines the energy (cost) for any two neighboring nodes to take the labels $L(x)$ and $L(y)$. As mentioned in Section 1, in an MRF setting this function is only a function of two labels - not a function of the node ids.

$$B(L(x), L(y)) = \text{Energy of neighboring nodes taking labels } L(x) \text{ and } L(y) \tag{4}$$

6 Message Passing

7 Initialization

As you will see in Section 8, to update the messages originating at a node, you must know all of the incoming messages to the node. This is a "chicken and the egg" problem. This is exactly what leads to the necessary modification from BP to Loopy BP. For standard BP (on a tree) you can traverse from the node to the leaves and always have visited “parent” nodes before all of their “children”, so this is not an issue. In fact, the order of traversal is well defined (which it is not in Loopy BP (see Section 8.2).

The modification to the algorithm which must be done to start Loopy BP is simply to initialize messages to 1.

8 Message update

To update each message from node i to node j, the following equation is used (see Section 8.2 for the order in which to update the messages).

$$m_{ij}(l) = \sum_{p \in L} \left[e^{-B(L(p),L(l))} e^{-U(L(p))} \prod_{k=N(i) \setminus j} m_{ki}(L(p)) \right] \tag{5}$$

It is from this equation that the algorithm gets its name - the “Sum-Product” algorithm. You can see that the message update equation (the main part of the algorithm) is the sum of a product. In the update equation, l is the particular label that the belief in the message pertains to. There are $|L|$ such messages between nodes i and j (i.e. $l \in |L|$).

We now explain each part of this equation.

- $m_{ij}(l)$

  This is the message from node i to node j containing information about node i's belief that node j is in state l.

- $\sum_{p \in L}$

  This is a summation over p (the variable name p was chosen arbitrarily) where p assumes all of the possible labels.

- $B(L(p), L(l))$

  This is the energy (recall from Section 5.2 that B is the name of the binary energy function) of neighboring nodes having labels $L(p)$ and $L(l)$.

- $U(L(p))$

  This is the energy (recall from Section 5.1 that U is the name of the unary energy function) of a node having label $L(p)$.

- $\prod_{k=N(i) \setminus j}$

  This is a product over k (the variable name k was chosen arbitrarily) where k assumes the values of the ids of all nodes neighboring node i except for node j.

- $m_{ki}(L(p))$

  This is an incoming message to node i from node k containing information about node k's belief of node i taking the value $L(p)$. (Note that if message $m_{ki}(L(p))$ was not computed before this update of $m_{ij}(l)$, there would be a problem. This is why the initializtaion to 1 (Section 7) is necessary.)

8.1 Normalization

As each message is pased, the set of messages between the pair of nodes should be normalized. Without this normalization, each time a message is passed the value of the message gets smaller (it is continually multiplied by a number less than 1), and eventually the value will become "0" to floating point precision. To fix this, we simply divide the value of each message from node i to node j by the sum over all of the values of the messages for all labels l:

$$m_{ij} \leftarrow \frac{m_{ij}}{\sum_l m_{ij}(l)} \tag{6}$$

8.2 Update Schedule

As noted in 7, there is no natural ordering of the nodes in an MRF. Because of this, you can update message in any order you would like. Often, a raster scan of the MRF is used. Alternatively, the message to update can also be selected at random.

8.3 Stopping Condition

The selected update schedule (Section 8.2) can be followed until your choice of stopping conditions is met:

- A fixed number of iterations
- Fewer than a specified number of messages are changed
- All messags are changed by an amount less than a specified threshold

9 Belief Computation

Once the selected stopping conditions (Section 8.3) have been met, the optimal labeling can be computed with

$$Belief(\text{node } i \text{ takes label } l) = U(l) \prod_{k=N(i)} m_{ki}(l) \tag{7}$$

That is, the belief that node i takes label l is the unary cost of label l ($U(l)$) multiplied by the product of the incoming messages of node i about label l ($\prod_{k=N(i)} m_{ki}(l)$).

10 Optimal Labeling

The optimal labeling of node i is determined simply by selecting the label l for which $Belief(\text{node } i \text{ takes label } l)$ is the largest.

11 Example Application - Binary Denoising

The binary denoising problem takes a binary image as input and outputs a binary image that has had the "noise removed". To solve this problem using the BP framework presented, we must simply specify appropriate cost functions.

11.1 Label set

In this problem, the label set L is $0, 1$, and hence $|L| = 2$. That is, every node (pixel) can either take the value 0 or 1 (a binary image).

11.2 Unary Cost

In this problem, the input image is called the "observations". The cost of assigning a node a particular label is related to if the label agrees with the observations. One such function is:

$$Unary(node, label) = \begin{cases} .2 & \text{if } observation(node) = label \\ .8 & \text{otherwise} \end{cases} \tag{8}$$

This function encourages the resulting labeling to be the same as the observations. If a node's label is the same as its original observation, the cost is .2. If a node's label is different from its original observation, the cost is .8.

11.3 Binary Cost

In a denoising problem, typically "smoothness" is the goal. That is, a node is likely to take the same label as its neighbors. The binary cost should encourage this smoothness. One such function to achieve this is:

$$Binary(label1, label2) = \begin{cases} 0 & \text{if } label1 = label2 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

This means if a node's label is the same as its neighbor, the cost is 0. If neighboring labels are different, the cost is 1.

12 Variants

13 Max-Product Algorithm

The max-product algorithm simply changes the message update equation from

$$m_{ij}(l) = \sum_{p \in L} \left[e^{-B(L(p),L(l))} e^{-U(L(p))} \prod_{k=N(i)\backslash j} m_{ki}(L(p)) \right] \tag{10}$$

(in the sum-product algorithm)
to

$$m_{ij}(l) = \max_{p \in L} \left[e^{-B(L(p),L(l))} e^{-U(L(p))} \prod_{k=N(i)\backslash j} m_{ki}(L(p)) \right] \tag{11}$$

14 Min-Sum Algorithm

The min-sum algorithm simply changes the message update equation from

$$m_{ij}(l) = \max_{p \in L} \left[e^{-B(L(p),L(l))} e^{-U(L(p))} \prod_{k=N(i)\backslash j} m_{ki}(L(p)) \right] \tag{12}$$

(in the max-product algorithm) to

$$m_{ij}(l) = \min_{p \in L} \left[B(L(p),L(l)) + U(L(p)) + \sum_{k=N(i)\backslash j} m_{ki}(L(p)) \right] \tag{13}$$

Note: With this algorithm, messages should be initialized to 0 instead of 1.