The Loopy Belief Propagation for Factor Graphs

Eingereichter Bericht
HAUPTSEMINAR NEURO-ENGINEERING
von

Caroline Blocher

geb. am 19. August 1990
wohnhaft in
Leonrodstraße 72
80636 München
Tel. 0176 27250499

Lehrstuhl für
STEUERUNGS- UND REGELUNGSTECHNIK
Technische Universität München

Univ.-Prof. Dr.-Ing. /Univ. Tokio Martin Buss

Betreuer: M.Sc. Indar Sugiarto

Beginn: 07.10.2014
Abgabe: 14.01.2015

Overview

This report treats Factor Graphs and Loopy Belief Propagation. Belief Propagation is a message passing algorithm which allows to efficiently perform inference on graphical models by exploiting the graph structure.

The first section presents the Factor Graph. The second section gives a brief introduction to Belief Propagation on tree-structured factor graphs. The third section provides an intuition on how to interpret Belief Propagation in terms of probabilistic inference.

The fourth section then extends the concept of Belief Propagation to cyclic graphs where the algorithm might not converge anymore. The fifth section shows some of the approaches that have been made to provide a better understanding of Loopy Belief Propagation. However, full comprehension of the algorithm on a general graph is not achieved yet according to the literature found within the scope of this report. On the one hand, empirical studies have been made, on the other hand some analytic analysis of message errors and conditions for convergence have been derived. A paper that analyzes the effects of message scheduling in Loopy Belief Propagation is presented as well.

The conclusion finally mentions some other algorithms that can be applied for cyclic graphs and discusses them briefly in comparison with Loopy Belief Propagation.

Contents

I Factor Graph

A factor graph is a graph representing the factorization of a global function g into a product of local functions f_s. The set of variables of each local function is a subset x_s of the variables of the global function g. Two kinds of nodes of the graph can be distinguished: Factor nodes and variable nodes. A factor node expresses a local function and is connected to each of the variable nodes representing a variable in the corresponding subset. Figure 1 shows an example of a factor graph and equation (1) the corresponding product of local functions. [2]

$$g(v,y,z,w) = f_A(v,y,z)\, f_B(y)\, f_C(z,w)\, f_D(z) = \prod_S f_s\ (x_s) \qquad (1)$$

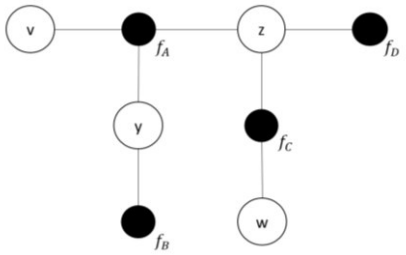

Figure 1: Example of a factor graph

II Sum-Product Algorithm (Belief Propagation) for tree-structured Factor Graphs

Belief Propagation is an application of the Sum-Product Algorithm. The algorithm is a message passing algorithm that sends messages from one node to another. In the following, the Sum-Product Algorithm for Factor Graphs as defined in [1] and [2] will be briefly presented.

The message a factor node f_s sends to a variable node x' is defined as:

$$\mu_{f_s \to x'}(x') = \sum_{x \in ne(f_s)\setminus x'} f_s(x,x') \prod_{x_m \in ne(f_s)\setminus x'} \mu_{x_m \to f_s}\ (x_m) \qquad (2)$$

One can see that first, the product of all incoming messages $\mu_{m \to f_s}(x_m)$ and the local potential $f_s(x,x')$ is calculated. This product is then marginalized over all variables that are direct neighbors of f_s except for x'. Hence, the message will only depend on x' as a variable.

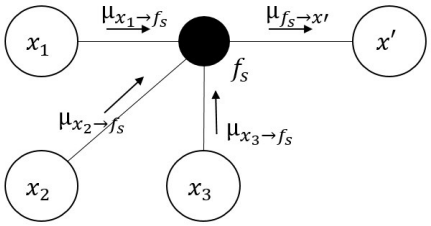

Figure 2: Outgoing message $\mu_{f_s \to x'}$ calculated at a factor node based on the incoming messages at all other links

The message a variable node x' sends to a factor node f_s is on the other hand defined as:

$$\mu_{x' \to f_s}(x') = \prod_{f_l \in ne(x')\setminus f_s} \mu_{f_l \to x'}\ (x') \qquad (3)$$

The variable node transmits the product over all incoming messages except the message from factor node f_s itself [1].

Frey, Kschischang, Loeliger and Wiberg describe in [2] how the messages are scheduled in the case of tree-structured networks. The first messages are sent from the leaf nodes to their neighbor. There is only one direct neighbor to a leaf node and therefore the leaf node does not need to compute the product of the incoming messages, but simply transmits its own local information. The neighbor then waits until it has received a message from all its leaf nodes, computes a message based on the incoming messages and sends it on the remaining edge. This is done until one message has been sent along each edge. If the tree has a single root node, it will then have incoming messages on all its edges. It therefore will capable to compute the corresponding outgoing messages for all its links and in the next iteration so will its children. The algorithm has finished if in both directions on each link a message has been sent.

The maximal number of iterations corresponds to the length of the longest path in the tree [4].

Figure 3 and 4 show the order of correct message sending in case of tree-structured networks. The numbers at the arrows correspond to the number of iteration. Factor node f_A and variable node z take both the place of the root node in the fourth iteration (see figure 4).

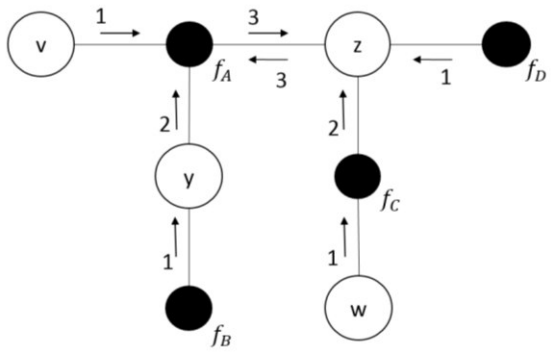

Figure 3: Message passing while no message has been sent on the link yet

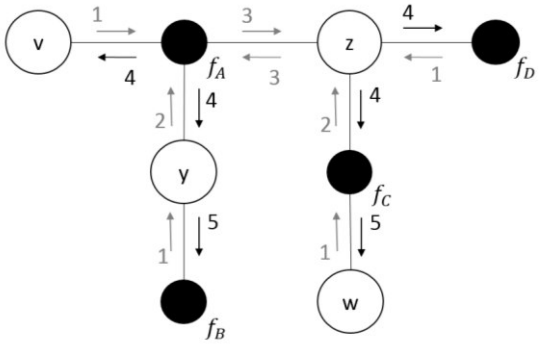

Figure 4: Message passing from root to leaves

III Inference

If the graph expresses a joint probability distribution by equation (1) and some of the nodes of the graph, for example the leaf variables, are fixed to an observed value, the posterior distributions of other nodes can be calculated by inference [1]. The following section wishes to remind of some background probability theory in order to provide an intuition on how to interpret the Belief Propagation Algorithm in terms of probabilistic inference.

The joint probability distribution of two dependent variables is given as:

$$p(x, y) = p(x)\, p(y|x) \qquad (4)$$

Suppose one would like to determine $p(x|y)$, the probability of x given the node y has been observed. It can be calculated by first marginalizing equation (4) over all x' (equation 5) and then using the Bayes' Theorem (equation 6):

$$p(y) = \sum_{x'} p(y|x')\, p(x') \qquad (5)$$

$$p(x|y) = \frac{p(y|x)\, p(x)}{p(y)} \qquad (6)$$

Defining a set of random variables ***x*** instead of {x, y} and with the general factor graph joint distribution being as in (1), the marginal distribution for the variable x' ∈ ***x*** is then:

$$p(x') = \sum_{\boldsymbol{x}\setminus x'} p(\boldsymbol{x}) \sim \sum_{\boldsymbol{x}\setminus x'} \prod_s f_s(x_s) \qquad (7)$$

The product of all incoming messages at variable node x' in Belief Propagation can hence be interpreted as the correct marginal for the variable x' [1] and is referred to as *belief* in the following sections.

IV Extension of Belief Propagation to general graphs

Belief Propagation can be applied to graph with cycles as well and has shown very good results in some cases. However, a theoretical understanding of the algorithm has not been provided yet except for the case of a single loop as mentioned in [7].

The main idea of the Loopy Belief Propagation algorithm remains the same as for the non-loopy case, except that messages along one edge have to be updated often based on different incoming messages at each iteration. Therefore, it is convenient to add a time index to each message μ. Message updates remain defined as in equations (2) and (3) but do not have to correspond to actual probabilities. They are positive functions and a normalization constant often is added as a factor [7].

The initialization and message scheduling of the algorithm must be defined in another way than for the tree case. In figure 5 one can see that neither variable node z nor factor node f_A are able to compute an outgoing message as there are not enough incoming messages. Therefore, instead of starting sending messages from the leaves, the messages on each link in every direction are initialized with a random value. Then each node can start computing outgoing messages thanks to its own local information and the initialized messages. The freshly computed outgoing message on a link will then replace the old one.

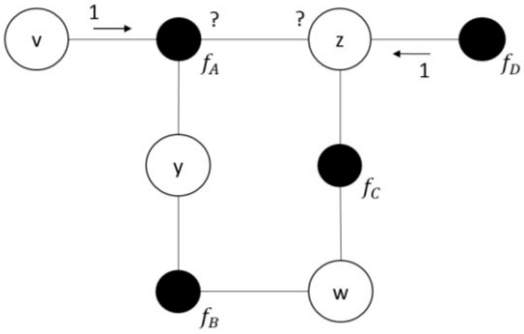

Figure 5: Example of loopy factor graph; Factor node f_A and variable node z cannot compute an outgoing message as there are not enough input messages

The message update rules can be defined in different ways. Bishop [1] mentions the flooding schedule, where all messages are updated simultaneously, and the serial schedule where only one message is updated at a time. In practice, the parallel update of all messages is used more often as it is easier to implement and as there are no clear rules yet on how to implement serial update [8]. In [8], however a proposition on how to implement asynchronous message passing is given (see section V-C).

One can see in figure 5 that the messages may be passed around the loop endlessly. However, the message values often converge after some iterations. Therefore, if analyzing the maximal difference between a message update and the previous message at some iteration i, one can define a threshold for this difference below which the algorithm can be considered converged. It may however occur in some cases that the message values oscillate and never converge [3].

In case of convergence, the result does not correspond the exact marginal, but research has shown that the approximation is often very good (see section V-A). Bolt and van der Gaag define in [5] two kinds of errors that occur if applying Belief Propagation to a Bayesian Network with loops. They help to give an intuition on why the result for Loopy Belief Propagation is only an approximate result.[1]

The first kind of error is the cycling error which arises due to the fact that messages are being passed around in the loop. If a node then computes an outgoing message he bases it on the incoming messages assuming they contain new information. However, due to the loop, some of the incoming messages will contain information he has sent himself during a previous iteration.
The second type of error called convergence error arises especially in the case of Bayesian Networks. A so-called convergence node, which is a node part of the loop, receives messages from its parents. When computing the standard outgoing message, it assumes that those messages are stochastically independent. This would however only be the case if the network was a tree-like network. As both its parents are part of the loop, they are in fact connected by another path beside the one passing the convergence node and therefore not independent [5].

V Some Investigations on Loopy Belief Propagation

According to the literature that has been found for this report a profound theoretical understanding of the Loopy Belief Propagation on a general graph has not been provided yet. Some of the issues that are still to investigate are for instance: How should the message passing method be chosen, when and why does Loopy Belief Propagation converge, and what is the quality of the result in this case. The following section presents some of the research approaches that have been done in these directions.

A - Empirical Approach

Murphy, Weiss and Jordan [3] have made an empirical approach and tested four different networks. Their experiences lead to the assumption that, if the algorithm converges, the results are usually close to the true marginal. If the algorithm oscillates, the correct marginal values seem to lay in the intervals defined by the oscillation, but this could not be shown for all cases.

Secondly, according to their results, not the structure of network but the parameter values seem to define whether one will observe convergence. The choice of the initial messages however does not matter for the convergence of the algorithm. [3]

B – Analysis of Convergence for Loopy Belief Propagation

Beside the empirical approach, some research has been done to derive analytical conditions for convergence of Loopy Belief Propagation.

These theoretical approaches often rely on concepts from statistical physics. Yedidia, Freeman, and Weiss [6] show that, if the Loopy Belief Propagation converges, the beliefs are then stationary point of the Bethe free energy. The local factors can be viewed as potentials. Bethe free energy can be defined as the sum of the free energies, calculated based on beliefs and potentials, of the clusters of nodes minus intersections. They also show that Belief Propagation always possesses a fixed point as the Bethe Free energy is bounded below. However, this fixed point is not necessarily unique and the algorithm does not necessarily converge to it.

[1] Directed or undirected graphs, i.e. Bayesian networks or Markov networks, can be transformed into a factor graph representation [1]. However, after a transformation the interpretation as in [5] might not be valid anymore.

In order to analyze the behavior of Loopy Belief Propagation, it is also helpful to use the concept of the computation tree. The main idea is that the effect of n iterations at a node s in the loopy graph is the same as the effect of exact belief propagation on the computation tree. The computation tree has as a root the node s. The paths from root s to the leaves are of length n and correspond to all possible movements from one node to its neighbors at the loopy graph during n iterations, starting at node s and not heading backwards directly (see figure 6). [4]

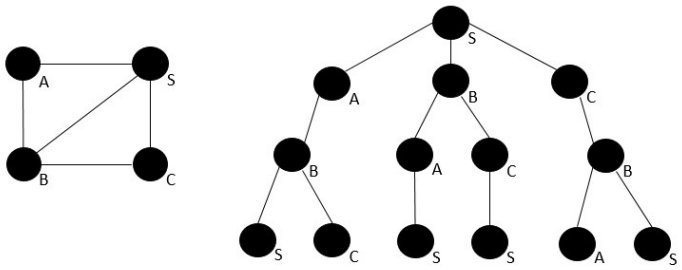

Figure 6: A loopy graph on the left hand side,
on the right hand side the corresponding computation tree unrolled from node s for 3 iterations

Ihler, Fisher and Willsky [4] analyze Loopy Belief Propagation by representing an approximate message m' as the product of the true message m and some error e. True messages correspond to a fixed point of Belief Propagation. They define a dynamic range measure d in order to measure the difference between two messages. This allows them to derive upper bounds on the error propagated and on the distance between any two fixed points. They are then able to derive the following sufficient condition in order to guarantee convergence of Loopy Belief Propagation based on the potentials of the graph:

$$max_{(s,t)\epsilon E} \sum_{u\epsilon\{ne(t)\setminus s\}} \log \frac{d(\Psi_{ut})^2-1}{d(\Psi_{ut})^2+1} < 1 \qquad (8)$$

For more details on this condition please refer to [4].

C - Choice of the message passing method

Elidan, McGraw and Koller analyze in [8] how to best schedule the messages. Experiences lead to the conjecture that asynchronous message passing works better for loopy networks although previous convergence analysis has mostly been done only parallel message updating. Elidan et al. show first that as one can assume that every message will at least be updated once within a finite interval of time, one can apply similar conditions for convergence as for parallel update. A comparison of parallel update with a round-robin schedule, where messages are updated serially in a predefined order, is done as well. They show that asynchronous (serial) message passing converges at least as fast as parallel updating.

Finally, they propose a message schedule called "Residual Belief Propagation". A residual is defined in this case as the difference between a message and its update. The main idea is to always update only the message with the largest residual, so the message whose update will have the biggest effect on the network. Experiments show that Residual Belief Propagation converges faster and more often than other approaches. [8]

VI Conclusion

One could raise the question on why use Loopy Belief Propagation if it has not been fully understood yet and if the result is only approximate. There exist for instance other algorithms which provide a solution for loopy networks.

The loopy network could for example be converted into a junction tree and then one could apply an algorithm corresponding to exact Belief Propagation. In practice, this solution is not efficient as the computational costs depend on the number of nodes. Real world networks tend to be large and the corresponding junction tree would be even larger [1].

A general comparison of Loopy Belief Propagation with other algorithm is difficult to do as there could not be derived any closed form solution for the algorithm yet. In the literature, some comparisons for certain real world applications can be found. For example, in [9], Loopy Belief Propagation is compared empirically to Mean Field Relaxation Labelling and Iterative Classification. In [3], Murphy et al. compare Loopy Belief Propagation to Likelihood Weighting with 200 samples. This is done empirically for three different networks where the algorithm converges and it is shown that in these cases, Loopy Belief Propagation gives better approximations for the marginal values than Likelihood Weighting.

In summary, there still remain several open issues and no general solution has been found yet in order to analyze Loopy Belief Propagation properly. However, Loopy Belief Propagation has shown to be very efficient in some real world applications [2]. So it is after all promising to do research in order to enhance the understanding of this algorithm.

References

[1] Christopher M. Bishop: “Pattern Recognition and Machine Learning”, p. 393-418, Springer, 2006.

[2] B. J. Frey, F.R. Kschischang, H.-A. Loeliger, N. Wiberg: “Factor Graphs and Algorithms” In IEEE Transaction on Information Theory, vol. 47, no. 2, 2001

[3] K. P. Murphy, Y. Weiss, M. I. Jordan: “Loopy Belief Propagation for Approximate Inference: An Empirical Study”, In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI1999), 1999

[4] A. T. Ihler, J. W. Fisher, A.S. Willsky: “Loopy Belief Propagation: Convergence and Effects of Message Errors”, In Journal of Machine Learning Research 6 pp. 905936, 2005

[5] J. H. Bolt, L. C. van der Gaag: “On the Convergence Error in Loopy Propagation”, Proceedings of the Third European Workshop on Probabilistic Graphical Models, Prague, pp. 43–50, 2006

[6] J. S. Yedidia, W.T. Freeman, Y. Weiss: “Generalized Belief Propagation” In NIPS 13, 689-695, 2000

[7] J. M. Mooij, H. J. Kappen: “Sufficient conditions for convergence of Loopy Belief Propagation”, In UAI, 2005

[8] G. Elidan, I. McGraw, D. Koller: “Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing”, In Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI2006), 2006

[9] P. Sen, L. Getoor: “Empirical Comparison of Approximate Inference Algorithms for Networked Data”, In A. Fern, L. Getoor and B. Milch (Eds.) Open Problems in Statistical Relational Learning: Papers from the ICML Workshop, Pittsburgh, 2006

[10] C. Axenie, J. Conradt Cortically inspired sensor fusion network for mobile robot egomotion estimation. Robotics and Autonomous Systems, Special Issue on “Emerging Spatial Competences: From Machine Perception to Sensorimotor Intelligence”, 2014, Volume 71, pages 69-82.