

MICHAEL RUDERER
BELIEF PROPAGATION

BELIEF PROPAGATION

MICHAEL RUDERER

Seminararbeit

Juni 2018 – version 0.0

Michael Ruderer: *Belief Propagation*, Seminararbeit, © Juni 2018

[June 10, 2018 at 20:48 – classicthesis version 0.0]

Ohana means family.
Family means nobody gets left behind, or forgotten.
— Lilo & Stitch

Dedicated to the loving memory of Rudolf Miede.
1939–2005

ABSTRACT

Short summary of the contents in English...a great guide by Kent Beck how to write good abstracts can be found here:

<https://plg.uwaterloo.ca/~migod/research/beck00PSLA.html>

ZUSAMMENFASSUNG

Kurze Zusammenfassung des Inhaltes in deutscher Sprache...

PUBLICATIONS

This might come in handy for PhD theses: some ideas and figures have appeared previously in the following publications:

Attention: This requires a separate run of bibtex for your refsection, e.g., `ClassicThesis1-b1x` for this file. You might also use biber as the backend for bibl2tex. See also <http://tex.stackexchange.com/questions/128196/problem-with-refsection>.

*This is just an early
– and currently
ugly – test!*

*We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.*

— Donald E. Knuth [1]

ACKNOWLEDGMENTS

Put your acknowledgments here.

Many thanks to everybody who already sent me a postcard!

Regarding the typography and other help, many thanks go to Marco Kuhlmann, Philipp Lehman, Lothar Schlesier, Jim Young, Lorenzo Pantieri and Enrico Gregorio¹, Jörg Sommer, Joachim Köstler, Daniel Gottschlag, Denis Aydin, Paride Legovini, Steffen Prochnow, Nicolas Repp, Hinrich Harms, Roland Winkler, Jörg Weber, Henri Menke, Claus Lahiri, Clemens Niederberger, Stefano Bragaglia, Jörn Hees, Scott Lowe, Dave Howcroft, and the whole L^AT_EX-community for support, ideas and some great software.

Regarding L_YX: The L_YX port was initially done by *Nicholas Mariette* in March 2009 and continued by *Ivo Pletikosić* in 2011. Thank you very much for your work and for the contributions to the original style.

¹ Members of GuIT (Gruppo Italiano Utilizzatori di T_EX e L^AT_EX)

CONTENTS

1	EINLEITUNG	1
2	BELIEF PROPAGATION	3
2.1	Factor Graphs	3
2.1.1	Factor graph of a SAT Problem	4
2.2	Message Passing Algorithms on Trees	4
2.3	Message Passing on general graphs	5
3	BELIEF PROPAGATION FOR SAT FORMULAS	7
3.1	Warning Propagation	7
3.1.1	Propagation Algorithm	7
3.1.2	Decimation Algorithm	8
3.2	Belief Propagation	9
3.2.1	Propagation Algorithm	9
3.2.2	Marginal Propabilities	9
3.2.3	Number of satisfying assignments	9
I	APPENDIX	
	BIBLIOGRAPHY	13

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

EINLEITUNG

Das ist die Einleitung

- Allgemeines über Message Passing, Belief Propagation
- Praktische Anwendungen (**TODO**: suchen)
- eventuell Motivation aus Quelle (hat mehr mit SAT als mit BP zu tun)
-

BELIEF PROPAGATION

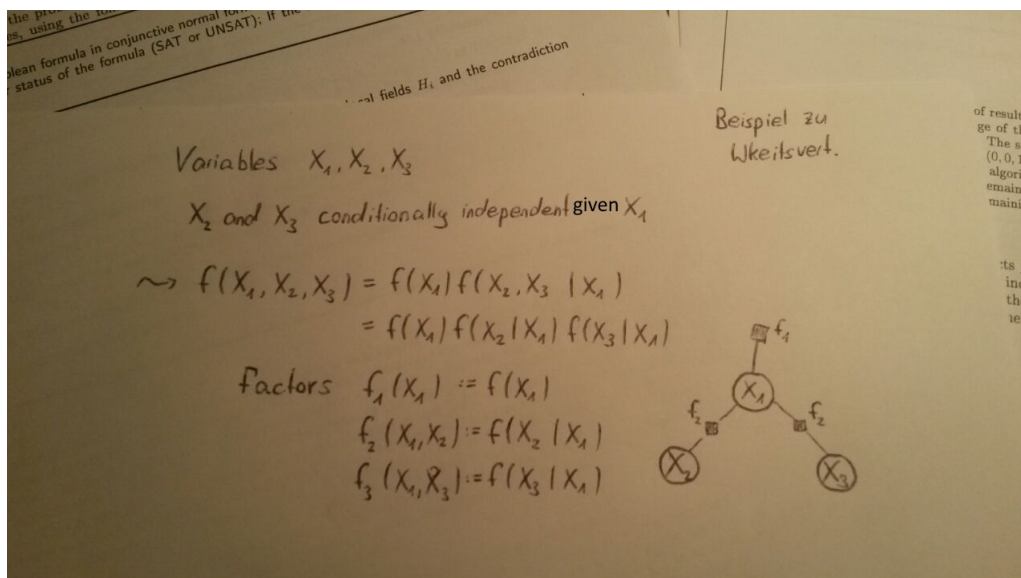
2.1 FACTOR GRAPHS

For problems that include many variables influencing each other it is useful to have an abstract representation of how those variables are related to each other. So called factor graphs are such representations.

In general, factor graphs represent the structure of a function's factorization into smaller functions.

If a function $f(X_1, \dots, X_n)$ can be written as a product $\prod_{j=1}^m f_j(S_j)$ where the functions f_j have smaller inputs $S_j \subset X$, its factorization can be expressed by a factor graph: The graph has two types of nodes: *variable nodes* that correspond to the variables X_i and *factor nodes* corresponding to the functions f_j . An edge connects a variable node X_i to a factor node f_j if X_i is part of f_j 's input. This means the factor graph is an undirected bipartite graph with the node set $V = \{X_1, \dots, X_n\} \cup \{f_1, \dots, f_m\}$ and edge set $E = \{(X_i, f_j) \mid X_i \in S_j\}$.

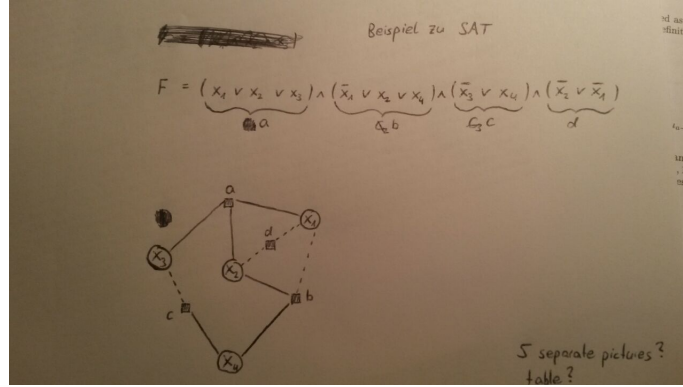
In many applications the global function f is a joined probability distribution that can be factorized by using information about independence between the variables. Typical tasks on factor graphs are computing variable assignments that maximize or minimize f or computing marginal distributions if f is a probability distribution. Both of these will be done in Section 3



Factor graphs can also be used for describing constraint satisfaction problems. A factor j corresponds to a constraint on its neighbour vertices, it evaluates to 1 if the constraint is satisfied and to 0 if not. For the global function f - the product of all factors - to be 1, every constraint has to be satisfied.

2.1.1 Factor graph of a SAT Problem

- definition: sat
- notation from source



In this example and in all following ones variable nodes are drawn as circles whereas constraint nodes are drawn as rectangles to easily distinguish the types of nodes. In the case of SAT an edge (x_i, C_j) is drawn dotted, if the variable appears in the clause C_j as \bar{x}_i , and continuous if not.

2.2 MESSAGE PASSING ALGORITHMS ON TREES

If the factor graph is a tree, many problems can be solved efficiently using a form of dynamic programming called *message passing*.

In general, message passing algorithms compute values for each edge of the factor graph. These values can be interpreted as *messages* that are sent between the nodes. Since all edges connect factor nodes to variable nodes there can be two types of messages: messages passed from a variable i to a factor a , denoted as $m_{i \rightarrow a}$ and messages passed from a to i , denoted as $m_{a \rightarrow i}$.

The messages must be defined so that a message $m_{a \rightarrow i}$ is determined by the messages $m_{j \rightarrow a}$ that a received from neighbour variables $j \neq i$. The same must hold for $m_{i \rightarrow a}$.

Usually the messages $m_{i \rightarrow a}$ are obtained by summing over $m_{b \rightarrow i}$ and $m_{a \rightarrow i}$ by multiplying the messages $m_{j \rightarrow a}$.

For tree factor graphs which do not contain cycles the value of $m_{j \rightarrow a}$ does not influence its predecessors $m_{b \rightarrow j}$. The messages can be computed sequentially starting with the factor graphs's leaves.

2.3 MESSAGE PASSING ON GENERAL GRAPHS

- "Loopy" belief propagation
- Convergence for trees
- Heuristic for general graphs
- Examples only in next chapter

In the following section two algorithms for SAT will be presented that both use the factor graph representation of a SAT formula. The description of both algorithms is based on [dummizitat].

3.1 WARNING PROPAGATION

The messages used in the Warning Propagation Algorithm (WP) presented in [[dummyzitat]] are called *warnings*. A warning $u_{a \rightarrow i} \in \{0, 1\}$ is passed from clause a to variable i . A converged warning $u_{a \rightarrow i}^*$ with value 1 should indicate, that to satisfy the clause a , the variable i has to take the value 1 if $j \in V_+(a)$ or 0 if $j \in V_-(a)$. The warning $u_{a \rightarrow i}^*$ will *fix* the variable i .

3.1.1 Propagation Algorithm

Like the general algorithm described in section 3.2.1 warning propagation is correct on trees and can be used as a heuristic for cyclic graphs by randomly initializing the warnings and hoping for convergence.

The algorithm starts by assigning each warning $u_{a \rightarrow i}$ a random starting value and updates these provisional warnings until their values have converged to a set of fixed point warnings $u_{a \rightarrow i}^*$ or until the number of iterations has exceeded some limit t_{max} .

The general idea is that the clause a has to fix the variable i only if the all of its other variables $j \in V(a) \setminus i$ are already fixed to values that do not satisfy the clause a .

The first step in the update procedure is to compute for each $j \in V(a) \setminus i$ the so called *cavity field* $h_{j \rightarrow a}$ that indicates what value j should take in the subproblem defined by $\tau_{j \rightarrow a}$. To compute $h_{j \rightarrow a}$ one has to count how many of the clauses $b \neq a$ fix j to 1 and how many fix j to 0:

$$h_{j \rightarrow a} = \sum_{b \in V_+(j) \setminus a} u_{b \rightarrow j} - \sum_{b \in V_-(j) \setminus a} u_{b \rightarrow j}$$

The clauses $b \in V_+(j)$ are the ones that would fix i to 1 if their warnings are active, the clauses $b \in V_-(j)$ would fix i to 0. So if $h_{j \rightarrow a}$ is positive, the variable i tends to the value 1, if the cavity field is negative it tends to 0. If $h_{j \rightarrow a} = 0$ which includes the case $V(j) \setminus a = \emptyset$ no conclusion can be made.

When all cavity fields are computed, each variable j with $h_{j \rightarrow a} \neq 0$ has a preferred value. This preferred value either makes the clause

a satisfied or does not contribute to the clause. If all variables $j \in V(a) \setminus i$ prefer a non satisfying value, the clause a sends a warning to i , meaning that i should take the satisfying value.

This warning can be computed by

$$u_{a \rightarrow i} = \prod_{j \in V(a) \setminus i} \theta(h_{j \rightarrow a} J_j^a) \quad \text{with } \theta(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{otherwise} \end{cases}$$

The factor $\theta(h_{j \rightarrow a} J_j^a)$ is 1 if j prefers to violate a and 0 if not:

If j has no preferred value its cavity field is 0 and $\theta(h_{j \rightarrow a} J_j^a) = 0$

If the preferred value of j satisfies a , J_j^a and $h_{j \rightarrow a}$ have different signs and $\theta(h_{j \rightarrow a} J_j^a) = 0$

If the preferred value of j violates a , J_j^a and $h_{j \rightarrow a}$ have the same sign and $\theta(h_{j \rightarrow a} J_j^a) = 1$

Warning Propagation Algorithm

0. Randomly initialize all warnings $u_{a \rightarrow i} \in_R \{0, 1\}$

1. For $t = 0$ to $t = t_{max}$

1.1 Compute in random order for all edges (a, i)

$$u_{a \rightarrow i} := \prod_{j \in V(a) \setminus i} \theta(-J_j^a \left(\sum_{b \in V(j) \setminus a} J_j^b u_{b \rightarrow j} \right))$$

1.2 If no message has changed goto 2.

2. If $t = t_{max}$ return UN-CONVERGED, else return the generated warnings $u_{a \rightarrow i}^*$

If the algorithm successfully returns a set of converged warnings, each warning $u_{a \rightarrow i}^* = 1$ fixes the variable i to the value satisfying a :

3.1.2 Decimation Algorithm

- Contradiction Numbers
- Lemma: $u_{a \rightarrow i}^* = 1 \Rightarrow i$ has a fixed value
- Decimation Algorithm
- Example on graph from last chapter (Je nachdem wie viel Platz in Tabellenform oder grafisch)

3.2 BELIEF PROPAGATION

Warning Propagation braucht bisher 2 Seiten, in der fertigen Version also etwa 4-5. Belief Propagation sollte in etwa den selben Umfang haben, etwas weniger Beschreibung nötig weil vieles gleich ist, dafür aber komplizierter.

Algorithmus In 3.2.1 Algorithmus beschreiben. Grundsätzliche Vorgehensweise ist dieselbe wie WP, also "nur" unterschiedliche Arten von Nachrichten und neue Update-Regel beschreiben

Lösungen Hier zeigen, wie die Anzahl der erfüllenden Belegungen und die Wahrscheinlichkeit für $x_i = 1$ aus den konvergierten Nachrichten berechnet wird. Wenn noch genug Seiten frei mit ausführlichem Beispiel

3.2.1 *Propagation Algorithm*

3.2.2 *Marginal Propabilities*

3.2.3 *Number of satisfying assignments*

Part I

APPENDIX

You can put some informational part preamble text here. Illo principalmente su nos. Non message *occidental* anglo-romanian da. Debitas effortio simplicate sia se, auxiliar summarios da que, se avantiate publicationes via. Pan in terra summarios, capital interlingua se que. Al via multo esser specimen, campo responder que da. Le usate medical addresses pro, europa origine sanctificate nos se.

BIBLIOGRAPHY

- [1] Donald E. Knuth. “Computer Programming as an Art.” In: *Communications of the ACM* 17.12 (1974), pp. 667–673.

DECLARATION

Put your declaration here.

, *Juni 2018*

Michael Ruderer

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst’s seminal book on typography “*The Elements of Typographic Style*”. `classicthesis` is available for both L^AT_EX and L^yX:

<https://bitbucket.org/amiede/classicthesis/>

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

<http://postcards.miede.de/>

Thank you very much for your feedback and contribution.