The max-sum Algorithm for Factor Graph

Hauptseminar
von

Dayou Wang

geb. am 01.10.1981
wohnhaft in:
Berg-am-laim-str.84
81673 München
Tel.: 0176 31245778

Lehrstuhl für
STEUERUNGS- und REGELUNGSTECHNIK
Technische Universität München

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

Betreuer: Betreuer: M.Sc. Indar Sugiarto
Beginn: 22.10.2012
Abgabe: 18.01.2013

Abstract

Graphical model is a natural tool used to evaluate the statistical relationships between the random variable exploiting the graph theory.In directed and undirected graph which are the two major classes of graphical model,a global function(joint distribution) can be expressed into a product of a local functions(marginals).Based on factor graph,the sum-product algorithm is studied,which provides a efficient algorithm to evaluate the local marginal distribution by using the idea of messages passing.Finally,a variation of sum-product algorithm—max-sum algorithm will be introduced,which calculate the maximal value of the joint distribution and the corresponding variables.

Zusammenfassung

Grafisches Model ist ein natürliches Werkzeug,das mit Ausnutzung der grafische Theorie verwendet wird zum Auswerten der statistischen Beziehungen zwischen die Zufallsvariablen.In gerichtetem Graph und ungerichtetem Graph, die zwei haupte Unterklasse von grafischem Model sind, eine globale Funktion(multivariate Verteilung) kann als ein Produkt von lokalen Funktionen(Marginalverteilungen) formuliert werden. Basierend auf Faktor Graph wird der sum-product Algorithmus einstudiert,der unter Verwendung einer Idee von messages passing ein effiziente Algorithmus zum Auswerten der lokale Marginalverteilung liefert.Schlieβlich wird eine Variante von sum-product Algorithmus—max-sum Algorithmus eingeführt,der den maximale Wert der multivariate Verteilung und die beziehende Variblen rechnet.

Contents

Chapter 1

Introduction

This seminar introduces the sum-product algorithm which operates in the factor graph. In a factor graph a global function(joint distribution) of a set of variables can be factorized into a product of local functions(marginal), each of that depends on a subset of variables.The sum-product algorithm provides an efficient computation to evaluate local functions using a message-propagating algorithm in a factor graph. Thereafter, the max-sum algorithm which is a variation of sum-product algorithm will be introduced to finde the maximum value of the global function and the variables that maximizes the global function.

Chapter 2

Graphical Models

Graphical models consist of nodes connected by links.Each node represents a random variable and a link probabilistic relationship between the variables.Usually there are two major classes of graphical models——directed graph and undirected graph.

2.1 Directed Graph

In the directed graph which is known as bayesian network,links carry arrows from parent nodes to children nodes.The joint distribution is written as a product of a conditional distribution over all of the nodes,each such conditional distribution is conditioned on the parent nodes of the corresponding node in the graph.

2.2 Undirected Graph

Undirected graph also called as markov network comprises a set of nodes which are connected by links without arrows. The joint distribution is defined as product of potential functions $\Psi_c(X_C)$ over the maximal cliques C of the graph,with the formel

$$p(X) = \frac{1}{z} \prod_C \Psi_c(X_C). \tag{2.1}$$

where Z is a normalization constant and defined by $Z = \sum_X \prod_C \Psi_c(X_C)$.

Chapter 3

Inference Problem

3.1 Problem Statement

Not all of the processes can be directly oberserved and discribed, but with the aid of some oberservable processes which have statistical relationship to the unoberservable processes,the truth of the unoberserved processes can be told. That is a inference problem. A classical example is that we can evaluate the weather by analyzing the moisture of the earth in a area. The main goal of inference problem is to evaluate the posterior distribution $p(x|y)$ using bayes' theorem $p(x|y) = \frac{(p(y|x)p(x))}{(p(y))}$, with the given conditional distribution $p(y|x)$ and prior distribution $p(x)$. The marginal distribution $p(y)$ is derived by marginalization of joint distribution $p(x, y)$ over all the x.Then the problem to calculate the posterior distribution is translated to be expressed in term of $p(x)$ and $p(y|x)$, which are already assumed.

3.2 Inference Problem On Markov Chain

In a more complex case involving the so-called Markov chain. The joint distribution in this undirected graph is given by

Figure 3.1:

$$p(X) = \frac{1}{z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)...\psi_{N-1,N}(x_{N-1}, x_N). \tag{3.1}$$

By definition, the marginal for the node x_n is the summation over all of the variable except x_n,

$$p(x_n) = \frac{1}{z}\sum_{x_1}...\sum_{x_{n-1}}\sum_{x_n}\sum_{x_{n+1}}...\sum_{x_N}p(X). \tag{3.2}$$

3.3 Messages And Message Propagating

To evaluate $p(x_n)$ we substitute the factorized expression for $p(X)$ into $p(x_n)$ and rerrange the summation and multiplication. We can do that because for example the term $\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$ is the only one in that the x_N appears. The desired

$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right] \cdots\right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots\right]}_{\mu_\beta(x_n)}.$$

marginal $p(x_n)$ can be splited in two parts $\mu_\alpha(x_n)$and $\mu_\beta(x_n)$ which can be viewed as a local message passed forwards and backwards along the chain. Rekursively, we have

$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi(x_{n-1}, x_n)\mu_\alpha(x_{n-1}) \tag{3.3}$$

$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi(x_{n+1}, x_n)\mu_\beta(x_{n+1}) \tag{3.4}$$

By definition the normalization constant Z is then obtained by summing $\mu_\alpha(x_n)\mu_\beta(x_n)$ over all states of x_n. This rekursive procedure is illustrated as below. This re-

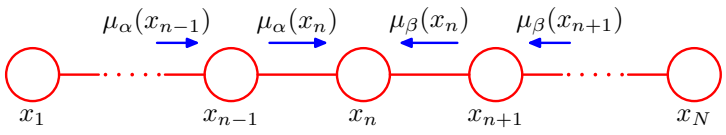


Figure 3.2:

arranging the order of summations and multiplications provides a foundation for sum-product algorithm.

Chapter 4

Tree And Factor Graph

4.1 Tree

The messages can be passed on a more complex structure called trees.A tree is defined as a graph in which there is only one link between any two nodes for an undirected graph.In the case of directed graphs a tree has a unique root node which has no parents,and all other nodes have one parent.

4.2 Factor Graph

Both undirected and directed trees can be converted to be a new graphical construction called factor graph,which consists of variable and factor nodes.A node is always connected by an undirected link to the opposite nodetype.
Compatibly,a joint distribution can be expressed as a product of factors over subsets of variables with formel $p(X) = \prod_s f_s(X_s)$,where f_s denotes the factor and X_s the set of variables,on which the factor depends.This factor is the local conditional distribution $p(x_k|pa_k)$ in directed graph and the potential function $\varphi_c(x_c)$ in undirected graph.The undirected links connect the factor node to all of the variable nodes on which this factor depends.Note that there are no loops in factor graph should be strictly restricted.

4.3 Convert To Factor Graph

To convert a undirected graph to a factor graph,we just create the variable nodes corresponding to the undirected graph and factor nodes corresponding to the potential function of maximal cliques,at last draw the appropriate links.In the case of directed graph,the variable nodes are created similarly and then factor nodes corresponding to conditional distributions are added.

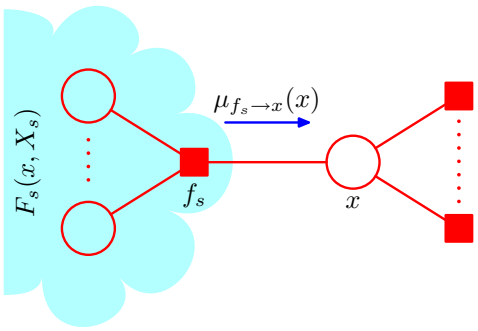
Chapter 5

The Sum-Product Algorithm

The sum-product algorithm is an efficient inference algorithm to calculate the local marginal $p(x)$ exploiting of tree structed factor graph.The tree structure of factor graph enable us to partition the factors into groups.One group is associated with each factor node f_s which is a neighbour of the variable node x.Then the joint distribution can be expressed as below,

$$p(X) = \prod_{s \in ne(x)} F_s(x, X_s) \tag{5.1}$$

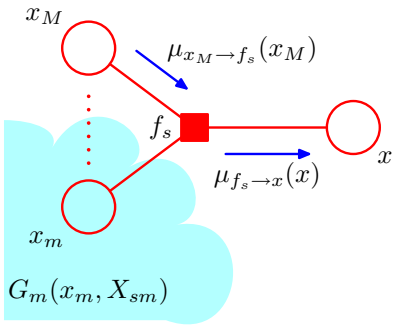
$ne(x)$ denotes the set of factor nodes which are neighnours of x and X_s the set of



all variable nodes connected to factor node f_s except x.$F_s(x, X_s)$ is the product of all the factors associated with f_s. Substituting (5.1) into $p(x) = \sum_{X/x} p(X)$ and rearranging summations and products we obtain

$$p(X) = \prod_{s \in ne(x)} \left[\prod_{X/x} F_s(x, X_s) \right] = \prod_{s \in ne(x)} \mu_{fs}(x). \tag{5.2}$$

Here $\mu_{fs}(x)$ is defined as messages from factor node to the variable node.Note that $F_s(x, X_s)$ is the product of factors and self can be factorized as The set x,x_1,x_2...x_M



$$\mu_{f_s \to x}(x) = \sum x_1 \ldots \sum_{x_M} f_s(x, x_1, \ldots x_M) \prod_{m \in ne(f_s) \setminus x} [\sum_{X_{sm}} G_m(x_m, X_{sm})]$$
$$= \sum x_1 \ldots \sum_{x_M} f_s(x, x_1, \ldots x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

is the set of variables on which the factor f_s depends. Then, where $ne(f_s)/x$ denotes the set of all the neighbour variable nodes of the factor node f_s with node x removed. Similarly to $\mu_{f_s \to x}(x)$, $\mu_{x_m \to f_s}(x_m)$ is the message from variable node to factor node and defined by

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm}) \tag{5.3}$$

$G_m(x_m, X_{sm})$ associated with node x_m in turn is the product of terms $F_l(x_m, X_{sm})$, each of which is associated with one factor node f_l ,which is its neighbouring factor excluding node f_s of node x_m ,so similarly we obtain

$$\mu_{x_m \to f_s}(x_m) = \prod_{lne(x_m)/f_s} (F_l(x_m, x_{ml})) = \prod_{lne(x_m)/f_s} \mu_{fl \to x_m}(x_m) \tag{5.4}$$

The computation beginns with designating a node as root and initiating the messages at each leaves of the tree.By definition of initialization,we obtain $\mu_{x \to f(x)=1}$ if the leaf node is a variable node ,and $\mu_{x \to f(x)=f(x)}$ if the leaf node is a factor node. Then we propagate the messages recursively until the messages are passed along every link and the root node has received messages from all of its neighbouring nodes.Once the message propagation is complete,we can then propagate messages from the root node out to the leaf nodes as before until every leaf node has received messages from all of its neighbouring nodes.Subsequently we can calculate a local marginal $p(x)$. To specify normalization constant Z, we firstly run the sum $-$ product algorithm to find the unnormalized marginals and Z is obtained by local computation over any one of these marginals.

Chapter 6

The Max-Sum Algorithm—Variation Of Sum-Product

The max-sum algorithm is used to find a set of variables who maximizes the joint probability and the value of this maximum probability. We define

$$X^{max} = arg_x max p(X) \tag{6.1}$$

$$P(X^{max}) = max_x p(X) = max_{x_1} ... max_{x_M} p(X) \tag{6.2}$$

Firstly we modify the sum-product algorithm to obtain the max-product algorithm by replacing the summation operator with maximum operator in messages $\mu_{f\to x}(x)$ and $\mu_{x\to f}(x)$. As before we arbitrarily choose a node as root and pass the messages from leaves to root until they all reach the root node. At the root we obtain the maximum probability by multipling all the incoming messages.

$$P^{max} = max_x \left[\prod_{s\in ne(x)} \mu_{f\to x}(x) \right] \tag{6.3}$$

Usually we prefer to use logarithm in scientific computation to avoid potential underflow. Because of the monotonic property of logarithm,what means if $a > b$ the $lna > lnb$, we can therefore interchange the max and the logarithm operators. This performing $ln(max_x p(X)) = max_x lnp(X)$ has a effect of replacing the products with sums. Therefore, we name this algorithm max-sum algrorithm. Finally, we obtain

$$P^{max} = max_x \left[\sum_{s\in ne(x)} \mu_{f\to x}(x) \right] \tag{6.4}$$

Now we should find out the maximizing variable set X^{max}. The most probable value for root x_N is then given by

$$x_N^{max} = arg_{x_N} max \left[\mu_{f_N\to x_N}(x_N) \right] \tag{6.5}$$

To determine the states of the previous variables of the maximizing configuration we use back-tracking, what is a application of dynamic programming. Each node has K possible states,for each state of a variable there is a unique state function involves it self and the previous variable which maximize the probability, defined by

$$\Phi(x_n) = arg\ max_{x_{n-1}} \left[lnf_{n-1,n}(x_{n-1}, x_n) + \mu_{x_{n-1}\to f_{n-1}}\ (x_n) \right]. \tag{6.6}$$

Beginning at the root node x_N we can then simply find the most probable state of previous node x_{N-1} using $x_{n-1}^{max} = \Phi(x_N^{max})$ and continue the operation back to the leaf node x_1 step by step. Finally we obtain the global maximizing configuration $x_1^{max} ...\ x_N^{max}$.

Chapter 7

Summary

Obviously, the sum-product algorithm has a great advantage at computation complexity than the navie calculation in finding a single marginal.
Suppose that there are N nodes in the chain each with K states.If we perform this summation explicitly to calculate $p(x_n)$,we obtain for N variables K^N values. This naive calculation with the exponential scale is very wateful and could cause a overload in computation. In sum-product algorithm we just need to computation twice the number of links in the tree, which saves a lot of computation time and resources.
It is also interesting to compare the sum-product algorithm and max-sum algorithm. To compute $p(x)$ with sum-product algorithm we need to sotre all of the messages from the neighbouring nodes of x towards x. And if we use max-sum algorithm, every time we perform a max operation, we should store the settings of the children nodes that led to the maximum. Because of its high computation efficiency,the max-sum algorithm is widely applied. For instance when applied in Hidden Markov Model,the max-sum algorithm is then known as viterbi algorithm.

Bibliography

[1] Frank R. Kschischang, Brendan J. Frey, Hans-Andrea Loeliger. *Factor Graphs and the Sum- Product Algorithm.*, In IEEE Transaction on Information Theory, vol. 47, no. 2, pp. 498-519, February. 2001.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning.*, Chapter 8. Springer. pp.359?418. ISBN 0-387-31073-8.