Belief propagation on graphs

Imen AYADI
ENS Paris-Saclay & CentraleSupélec
imen.ayadi@student.ecp.fr

Zaineb LETAIF
CentraleSupélec
zaineb.letaif@student.ecp.fr

Abstract

Belief propagation (BP) is an increasingly popular method of performing approximate inference on graphical models. However at times the inherent structure of the graph can be an obstacle to BP convergence. Hence the need to Generalize BP algorithm to adapt it to arbitrary graphs specifically in the case of cycles. In this project we investigate the effectiveness of loopy BP through a number of experiments.

1. Introduction

Probabilistic graphical models are a rich framework for encoding probability distributions over complex domains: joint (multivariate) distributions over large numbers of random variables that interact with each other. These representations are the basis for the state-of-the-art methods in a wide variety of applications, such as medical diagnosis, image understanding, speech recognition, natural language processing,etc. Thus, it is important to derive methods for the probabilistic inference. There are both exact and approximate algorithms for different types of inference tasks; choosing the best algorithm depends merely on the structure of the graph. One of these inference algorithms is Belief propagation (BP) which is an iterative process in which neighboring variables talk to each other and after enough iterations, this series of conversations is likely to converge to a consensus that determines the marginal probabilities of all the variables. Estimated marginal probabilities are called beliefs. In Section 3, we will present the BF algorithm for trees. While BF is an exact inference toll for non-cyclic graphs, it could be adapted for graphs with loops to bring . In Section 4, we focus on this adapted version of BP, called loopy belief propagation (LBP) and how to implement it. In Section 5, we compare through experiments the performance of LBP with other inference techniques.

2. Notations

Let $G = (V, E)$ denotes a graph with nodes $V = \{x_1, ..., x_n\}$ and edges E. For a given node x_i, we note $\mathcal{N}(x_i)$ the set of neighbors of x_i.
We restrict our study on the common case of **pairwise MRF**, which consists on just considering only unary and pairwise factors. Thus,

$$P(x_1, ..., x_n) = \frac{1}{Z} \prod_{i=1}^{n} g_i(x_i) \prod_{i<j} f_{ij}(x_i, x_j) \qquad (1)$$

where g_i refers to a unary factors and f_{ij} to a pairwise factor.
Let $m_{i \longrightarrow j}(x_j)$ the message from the node x_i to the node x_j, a function of the states of the variable x_j.

3. Belief Propagation on trees

The BP [4] consists on message passing between clusters each of which encodes a factor over a subset of variables. In fact, a node can send a message to its neighbors when

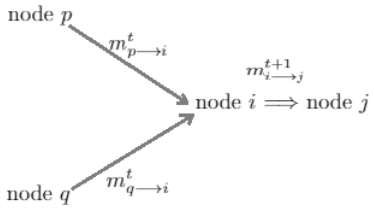


Figure 1: Message Updating

and only when it has received message from all its other neighbors. The message is defined as:

$$m_{i \longrightarrow j}(x_j) = \sum_{x_i} f_{ij}(x_i, x_j)\, g(x_i) \prod_{x_k \in \mathcal{N}(x_i) \setminus \{x_i\}} m_{k \longrightarrow i}(x_i) \qquad (2)$$

The node marginal can be calculated by multiplying all the messages with its own potential (when it receives all the

messages from its neighbours) as given by:

$$p(x_i) \propto g(x_i) \prod_{x_k \in \mathcal{N}(x_i)} m_{k \longrightarrow i}(x_i) \tag{3}$$

- **Implementation:** In the first step, starting at the leaves , each node passes a message along the unique edge towards the root. Once the root has received messages from all its direct children, we pass messages back out: starting at the root, messages are passed in the reverse direction and the algorithm finishes once all the leaves have received their messages. We use dynamic programming to code the BP algorithm.

- **Computational Complexity:** If nothing were known about the joint probability structure, the marginalization cost would grow exponentially with the number of nodes in the network. In fact, if each node x_i has k possible states, then, the algorithm require $O(k^n)$ computations. However,since the graph structure corresponding to the joint probability is a tree, then the marginalization cost only grows linearly with the number of nodes, and is quadratic in the node state dimensions.

4. Belief Propagation on general graphs

Algorithms based on message passing perform well on trees and chain. However, for loopy graphs (graph that contains loops), messages may circulate indefinitely around the loops and hence may not converge [3]. Thus, the full algorithm is given by 1:

Algorithm 1: Loopy Belief Propagation Algorithm

Result: $p(x_i)$ for $x_i \in V$

initialization : all messages $\mu_{i \to j}^{(0)} = 1 \; \forall (i,j) \in E$

for $t \in \{1, \cdots, t_{max}\}$ **do**

$\forall (i,j) \in E$, compute:

$m_{i \longrightarrow j}^{(t)}(x_j) =$

$\sum_{x_i} f_{ij}(x_i, x_j)\, g(x_i) \prod_{x_k \in \mathcal{N}(x_i) \setminus \{x_i\}} m_{k \longrightarrow i}^{(t-1)}(x_i)$

- compute beliefs:
  $p(x_i) = \frac{1}{Z} \prod_{j \in \mathbf{N}(x_i)} m_{j \to i}^{(t_{max}+1)}(x_i)$
- normalize $p(x_i)$

Compared with the algorithm for trees, we modified the initialization step.
The message update rules are no longer guaranteed to return the exact marginals, however LBP fixed-points correspond to local stationary points of the Bethe free energy [1] : that is why if we have convergence, then, we are sure that our approximation of marginal distributions is good.

5. Experiments

We run some experiments using our implementation of Loopy Belief Propagation and we compare the results with the ones given by other inference methods. All the parts of the code are available in https://github.com/IA3005/Graph_models_inference.git.

5.1. Experiment 1: Simple graph with a cycle

This first experiment serves to verify if our implementation of LBP works well. Let's consider the factor graph with a loop in Figure 2 Our goal is to compute marginal

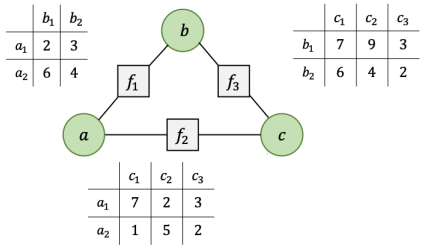


Figure 2: A loop graph

distribution of b using LBP.
Firstly, using marginalization, we compute exact marginal distribution of $b^* = (b_1^*, b_2^*)$ and normalize it. Now, we run the LBP and we plot the evolution of the inference error $||b_k - b^*||_2$ as a function of the number of iterations k used in LBP. The plot is provided in Figure 3

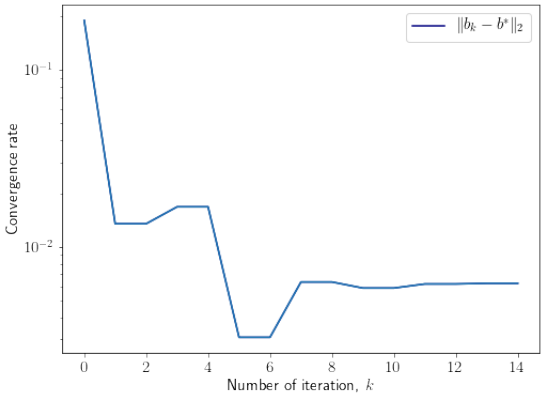


Figure 3: Evolution of the inference error $||b_k - b^*||_2$ as a function of the number of iterations k used in LBP

5.2. Experiment 2: Ising model for image denoising

We consider a simple version of the Ising model [1]:

$$p(x) = \frac{1}{Z_{\alpha,\beta}} exp\left\{ \alpha \sum_{i=0}^{n-1} x_i + \beta \sum_{(i,j)\in E} \mathbf{1}(x_i = x_j) \right\} \tag{4}$$

where x_i are either 0 or 1 and α, β are fixed parameters. We use this model to remove noise from a gray-scale image.
We do not observe x directly. We instead observe independant variables y_i which, conditional on $x_i = l \in \{0, 1\}$ are distributed according to a Gaussian distribution $\mathcal{N}(\mu_l, 1)$.
To model the problem with a graph, each node represents a pixel of the grayscale image (then $n = H \times W$ is the number of nodes of the considered graph where H and W are the dimensions of the image). A node x_i could be linked to its 4 direct neighbors if they have the same color.
We recall that a neighbor is immediately located on the left, right, up or down of the considered pixel.
The Ising model is well suited for the task of image denoising. We take each observable nodes y to correspond to a pixel in the noisy observed image and the pixels of the "true state" image with noise removed to correspond to the latent nodes x. If we marginalize each x_i and take the most likely belief for each to be our estimate of each true pixel value, then we can reconstruct the image with these estimates as our overall "denoised" image (see the graphical model in Figure 4.

In fact, the maximal cliques in the model are the edges in

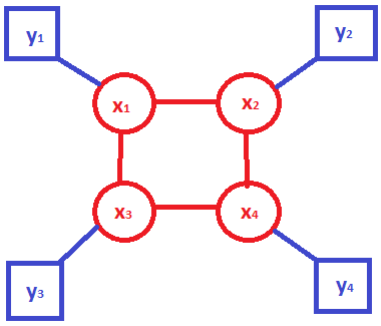


Figure 4: Graphical model for denoising: y_i are observable and x_i are hidden

the graph, and there are 2 types of maximal cliques. The first type of maximal clique connects the denoised image (unobserved) nodes $\{x_i, x_j\}$ and represents the relationship between neighbouring pixels. The second type of maximal clique connects denoised image nodes to noisy image nodes $\{x_i, y_i\}$ and represents the noise.
We can compute the marginals of a large set of random variables in an MRF using belief propagation. since the considered graph is not a tree, we use the loopy version.
Let $\{v_{i,j}|(i,j) \in [\![0, H-1]\!] \times [\![0, W-1]\!]\} := \{x_k | k \in [\![0, n-1]\!]\}$ with $k = i \times W + j$. Then,

$$\begin{aligned} &\prod_{k=0}^{n-1} \exp(\beta \sum_{(k,l)\in E} \mathbf{1}(x_k = x_l)) \\ &= \left[\prod_{i=0}^{H-1} \prod_{j=0}^{W-2} e^{2\beta \mathbf{1}(v_{i,j} = v_{i,j+1})} \right] \left[\prod_{i=0}^{H-2} \prod_{j=0}^{W-1} e^{2\beta \mathbf{1}(v_{i,j} = v_{i+1,j})} \right] \end{aligned} \tag{5}$$

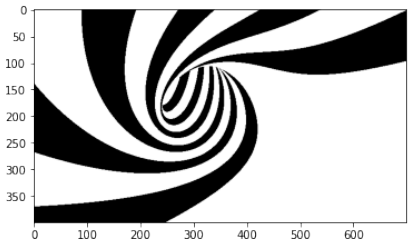
Therefore, the joint distribution is given by:

$$\begin{aligned} &p(x_0, \dots, x_{n-1}) \\ &= \frac{1}{Z_{\alpha,\beta}} \left(\prod_{i=0}^{H-1} \prod_{j=0}^{W-1} e^{\alpha v_{i,j}} \right) \underbrace{\prod_{i=0}^{H-1} \prod_{j=0}^{W-2} e^{2\beta \mathbf{1}(v_{i,j} = v_{i,j+1})}}_{\text{horizontal edges}} \\ &\quad \times \underbrace{\prod_{i=0}^{H-2} \prod_{j=0}^{W-1} e^{2\beta \mathbf{1}(v_{i,j} = v_{i+1,j})}}_{\text{vertical edges}} \\ &= \frac{1}{Z_{\alpha,\beta}} \prod_{i=0}^{H-1} \left(\prod_{j=0}^{W-1} e^{\alpha v_{i,j}} \prod_{j=0}^{W-2} e^{2\beta \mathbf{1}(v_{i,j} = v_{i,j+1})} \right) \\ &\quad \times \prod_{i=0}^{H-2} \left(\prod_{j=0}^{W-1} e^{2\beta \mathbf{1}(v_{i,j} = v_{i+1,j})} \right) \\ &= \frac{1}{Z_{\alpha,\beta}} \prod_{i=0}^{H-1} \phi_i(x_{R_i}) \prod_{i=0}^{H-2} \phi_{i,i+1}(x_{R_i}, x_{R_{i+1}}) \end{aligned}$$

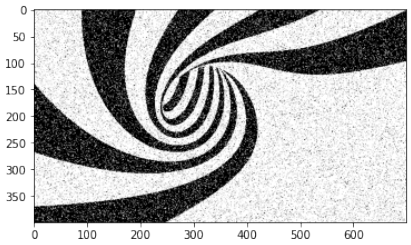
Now, since we identified our factors, we can apply the LBP as described in Section 4. To assess the performance of such algorithm, we decided to compare it with another inference technique called Gibbs Sampling (or simply MCMC)[2]: we denoise the same input image with LBP and Gibbs Sampling and we compare the MSE of the denoised images with respect to the noiseless input. We also implemented the simulated EM algorithm we combine the MCMC with the Expectation-Minimmization EM algorithm to denoise the image and learn at the same time μ_0 and μ_1 to extend the comparison. The result of denoising a image of spiral is provided in Figure 5. We notice that LBP is obviously faster than Gibbs sampling and provides a more denoised image (less MSE). Besides, Gibbs is very sensible to the number of iterations.

[1] This experiment was inspired from an assignment of the course Probabilistic Graphical Models (MVA) taught by Pierre Latouche and Nicolas Chopin

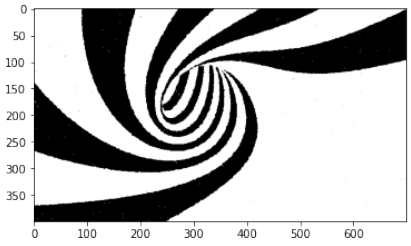
[2] $P(x_k|x_{\neg k}, y) = \frac{p(x_k|y)}{p(x_{\neg k}|y)} \propto \phi_k(x_k) \prod_{l\in\mathcal{N}(k)} \mu_{l\to k}(x_k) \mathcal{N}(\mu_{x_k}, 1) f(x_{\neg k})$



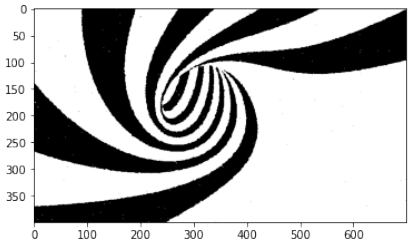
(a) Noiseless image



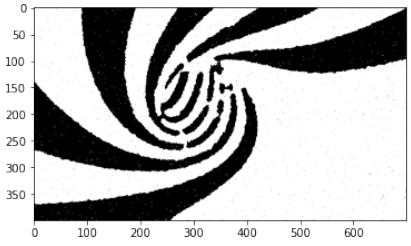
(b) Noisy image : we added a salt and pepper noise with probability $p = 0.1$ to the original image



(c) Denoised image with LBP (MSE = 48.1)+ fixed $\mu_0 = 0$ and $\mu_1 = 1$



(d) Denoised image with Gibbs sampling (MSE = 87.58)+ fixed $\mu_0 = 0$ and $\mu_1 = 1$



(e) Denoised image with Simulated EM (MSE = 58.3)+ μ_0 and μ_1 are learned by the algorithm : we find $\mu_0 = 0.12$ and $\mu_1 = 0.88$

Figure 5: Denoising an image with the Ising model with fixed parameters ($\alpha = 0.03$ and $\beta = 0.6$)

5.3. Experiment 3: Stereo Matching

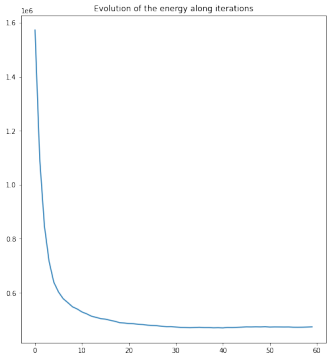
The purpose of this experiment is to compute the disparity between two images (from which can be recovered the depth of field of the objects in the image). The input are two images of the same scene taken from two different points. We assumed that the two points differ only by an horizontal translation.
The problem of recovering an accurate disparity map L can be posed as an energy minimization problem:

$$\mathcal{E}(L) = \sum_{p \in \mathcal{P}} D_p(l_p) + \lambda \sum_{p,q \in \mathcal{N}} V(l_p - l_q)$$

where $\mathcal{P}$ is the set of pixels, $\mathcal{N}$ is the set of undirected edges in the four-connected image grid graph and $l_p \in \{0, .., d_{max}\}$ is the disparity value of a pixel p (such that d_{max} is the maximum depth).
$D_p(l_p)$ is the cost of assigning the label disparity l_p to pixel p. It can be modeled with the truncated absolute intensity difference :

$$D_p(l_p) = min\left(\frac{1}{3}|I_{\text{left}}(y,x) - I_{\text{right}}(y, x - l_p)|, \tau\right)$$

where τ is a given threshold .
$V(l_p - l_q)$ is the cost of assigning labels l_p and l_q to two neighbor pixels p and $q \in \mathcal{N}$. With the Potts model, $V(x) = \mathbf{1}_{\{0\}}$. We implement a solution for this MAP using Loopy Belief Propagation. The results are plotted in Figure 6: We compare the LBP solution with



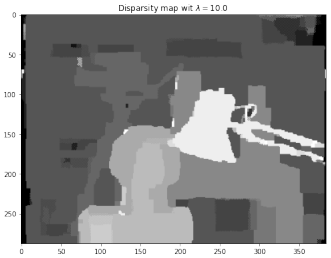


Figure 6: Stereo Matching with LBP : evolution of the energy as function of iterations + the disparity map / Parameters : $d_{max} = 16$, $\tau = 15$, $\lambda = 10$

the one provided by the graph cuts: in fact, we use the code provided in https://github.com/sjawhar/cv-stereo-disparity-graph-cuts.git which implemented the Kolmogorov and Zabih's Graph Cuts Stereo Matching Algorithm [2]. We notice that LBP's disparity map is closer to the gound-truth one than the disparity map provided by the graph cuts. Maybe, if we choose a more charged picture, we could have a lower accuracy of the LBP solution.

6. Conlusion

In This project We took interest in the efficiency of loopy BP for resolution of inference problems when graph structure is arbitrary and can contain loops. We proposed 3 experiments to test the efficiency of our implementation and compared results with other algorithms suitable for the inference tasks. All in all loopy BP is a good option when considering loop graphs and allows to avoid non-convergence issues.

References

[1] B. Freeman and A. Torralba. Lecture 7: Graphical models and belief propagation.

[2] Z. Kolomogorov. Kolmogorov and zabih's graph cuts stereo matching algorithm.

[3] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study.

[4] J. S.Yedida and W. T.Freeman. Understanding belief propagation and its generalizations.