Efficient Implementation of Belief Propagation on Graphs

Zaineb Letaif , Imen Ayadi

Discrete inference on Graphical Models Project

31 March 2021

Outline

Introduction

- Probabilistic Graphical Models: A rich representation that is basis for multiple state-of-the-art methods.
- Algorithms to infer information from graphs vary depending on type of task and structure of graph.
- For this project we take interest into The *Belief Propagation (BP)* Inference algorithm

Introduction

- Probabilistic Graphical Models: A rich representation that is basis for multiple state-of-the-art methods.
- Algorithms to infer information from graphs vary depending on type of task and structure of graph.
- For this project we take interest into The *Belief Propagation (BP)* Inference algorithm

Introduction

- Probabilistic Graphical Models: A rich representation that is basis for multiple state-of-the-art methods.
- Algorithms to infer information from graphs vary depending on type of task and structure of graph.
- For this project we take interest into The *Belief Propagation (BP)* Inference algorithm

Objectives

- For our implementations we take interest into a graph $G = (V, E)$ an We restrict our study on the common case of **pairwise MRF**.
- Firstly we consider the Belief Propagation method applied to tree graphs.
- In a second time we will present a *BP* method adapted to graphs containing loops: called *Loopy Belief Propagation* its implementation and different application examples.

Problem Definition

- We consider a Graph $G = (V, E)$ with nodes $V = \{x_1, ..., x_n\}$ for each node x_i $N(x_i)$ is the set of its neighbours.
- Pairwise MRF:

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

- Message Passing: the message from the node x_i to the node x_j is defined as:

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

Problem Definition

- We consider a Graph $G = (V, E)$ with nodes $V = \{x_1, ..., x_n\}$ for each node x_i $N(x_i)$ is the set of its neighbours.
- Pairwise MRF:

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

- Message Passing: the message from the node x_i to the node x_j is defined as:

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

Problem Definition

- We consider a Graph $G = (V, E)$ with nodes $V = \{x_1, ..., x_n\}$ for each node x_i $N(x_i)$ is the set of its neighbours.
- Pairwise MRF:

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

- Message Passing: the message from the node x_i to the node x_j is defined as:

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

Implementation

- BP algorithm is implemented using dynamic programming: Viterbi.
- **Computational Complexity:** graph structure corresponding to the joint probability is a tree $\Rightarrow$ marginalization cost only grows linearly with the number of nodes
- If joint probability unknown then cost grows exponentially with number of nodes

Challenges

- Graphs containing cycles can lead to messages circulating indefinitely.
- In this case classic *BP* implementation is not applicable as it may not converge.
- Arbitrary Graph structure is at times a bottleneck for classic *BP*.

Methodology

Loopy algorithm

Algorithm 1: Loopy Belief Propagation Algorithm

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

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

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

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

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

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

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

Methodology

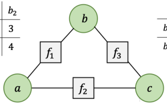
Loopy algorithm

- Modified initialization step: all messages $\mu^{(0)}_{i \to j} = 1 \ \forall (i,j) \in E$
- Message updates do not return exact marginals
- LBP fixed-points correspond to local stationary points of the Bethe free energy $\Rightarrow$ This guarantees convergence.

Results

Simple graph containing cycle

	b_1	b_2
a_1	2	3
a_2	6	4



	c_1	c_2	c_3
b_1	7	9	3
b_2	6	4	2

	c_1	c_2	c_3
a_1	7	2	3
a_2	1	5	2

Figure: A simple loop graph

- **Goal:** compute marginal distribution of b using LBP to verify our implementation.

Results

Simple graph containing cycle

- 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.

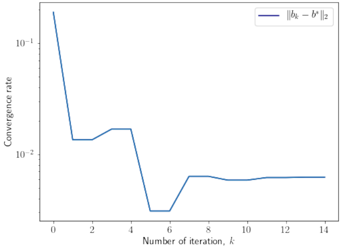


Figure: A simple loop graph

Results

Image Denoising : Ising model

- **Ising model:** A simple version

$$p(x) = \frac{1}{Z_{\alpha,\beta}} exp \left\{ \alpha \sum_{i=0}^{n-1} x_i + \beta \sum_{(i,j)\in E} \mathbf{1}(x_i = x_j) \right\} \qquad (3)$$

Results

Image Denoising : Ising model

- **Image denoising as a graph**

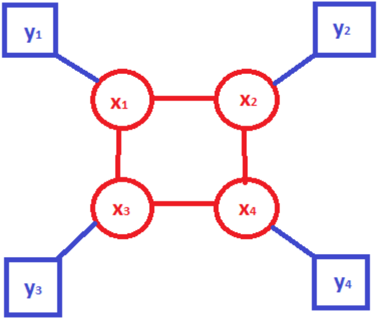


Figure: Graphical model for denoising: y_i are observable and x_i are hidden

Results

Image Denoising: Ising model

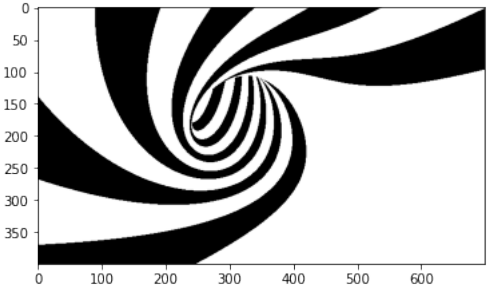


Figure: Noiseless image

Results

Image Denoising: Ising model

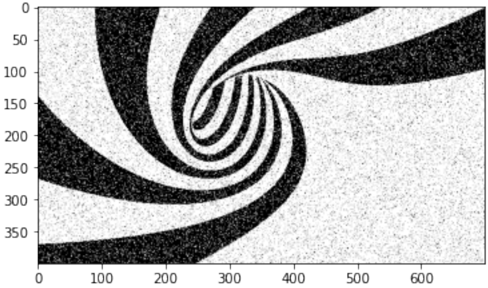


Figure: Noisy image : we added a salt and pepper noise with probability $p = 0.1$ to the original image

Results

Image Denoising: Ising model

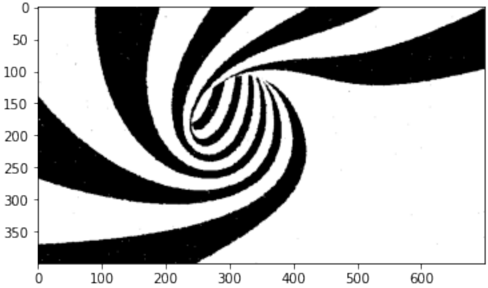


Figure: Denoised image with LBP (MSE = 48.1)+ fixed $\mu_0 = 0$ and $\mu_1 = 1$

Results

Image Denoising: Ising model

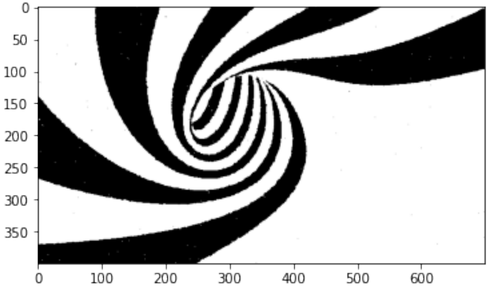


Figure: Denoised image with Gibbs sampling (MSE $= 87.58$)+ fixed $\mu_0 = 0$ and $\mu_1 = 1$

Results

Image Denoising: Ising model

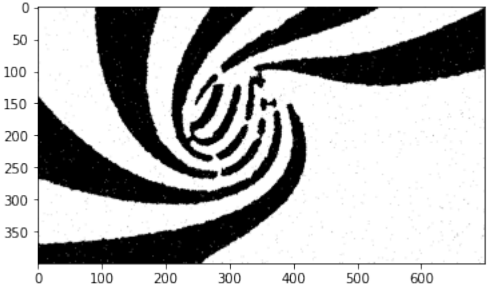


Figure: Denoised image with Simulated EM (MSE $= 58.3$)+ μ_0 and μ_1 are learned by the algorithm : we find $\mu_0 = 0.12$ and $\mu_1 = 0.88$

Results

Stereo Matching

- **Goal:** Calculate disparity between two images and thus recuperate depth of filed of objects in the image.

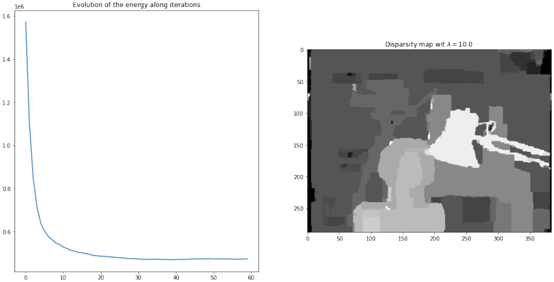


Figure: Stereo Matching with LBP : evolution of the energy as function of iterations + the disparity map / Parameters : $d_{max} = 16$, $\tau = 15$, $\lambda = 10$

Evaluation of the results

- Loopy BP on simple loop graph: Our implementation of LBP is successful. The estimated distribution converges towards ground truth distribution with no bottleneck problems.
- Image Denoising: LBP has very good visual results with low MSE (48.1).
  Compared to other algorithms:Gibbs sampling and EM it shows comparable numeric performance results (especially with EM (MSE = 48.1)) while being also a good reconstitution of the ground truth image.