Markov Random Fields and Gibbs Sampling for Image Denoising

Chang Yue
Electrical Engineering
Stanford University
changyue@stanfoed.edu

Abstract

This project applies Gibbs Sampling based on different Markov Random Fields (MRF) structures to solve the image denoising problem. Compared with methods like gradient ascent, one important advantage that Gibbs Sampling has is that it provides balances between exploration and exploitation. This project also tested behaviors of different MRF structures, such as the high-order MRFs, various loss functions when defining energy, different norms for sparse gradient prior, and the idea of 'window-wise product'. I finally generated an output which is an ensemble of many structures. It is visually good, and comparable to some state-of-art methods.

1. Introduction

Image denoising has been a popular field of research for decades, and lots of methods have been developed. A typical approach would be going through an image pixel by pixel, and, at each pixel, by applying some Math models, set that pixel's value based on the evidence of its neighbors' values. Gaussian smoothing is the result of blurring an image by a Gaussian function, typically to reduce image noise. Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function [2]. Gradient ascent is another widely used method, where it iteratively assigns pixels values such that they bring improvement on the pre-defined function (usually called loss function). It will converge to a suboptimal state, and different initializations give different results in general. The drawback of these methods is that they are too deterministic: assign pixel value based on the current state with a probability of 1. And, methods like gradient ascent need careful initialization.

Gibbs Sampling is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution [4]. It provides exploration by sampling a pixel's value based on probabilities computed from a MRF model. Therefore, a seemingly high-value pixel might find itself improves the whole picture's loss after it got a chance to be sampled to a low-value. It is proved that after enough iteration passes, Gibbs Sampling will converge to the stationary distribution of the Markov Chain, no matter what the initialization is. This is shared by all Metropolis-Hastings algorithms, but Gibbs Sampling, a special case of Metropolis-Hastings, has a big advantage in image denoising applications, when the conditional distribution of each variable is known and is easy to sample from.

Markov Random Field (MRF) is a set of random variables (i.e., pixel values) having a Markov property described by an undirected graph [8]. It represents undirected dependencies between nodes. For a single node, we can compute its probability distribution given evidence of its neighbors (named Markov Blanket). Different MRF will yield different distributions, and therefore, different sampling results. The second part of this project is to explore the effectiveness of different MRF.

2. Related Work

Markov Random Field Models (MRF) theory is a tool to encode contextual constraints into the prior probability [5]. S. Z. Li presented an unified approach for MRF modeling in low and high level computer vision. Roth and Black designed a framework for learning image priors, named fields of experts (FoE) [7]. McAuley *et al*. used large neighborhood MRF to learn rich prior models of color images, which expands the approach of FoE [6]. Barbu observed considerable gains in speed and accuracy by training the MRF/CRF model together with a fast and suboptimal inference algorithm, and defined it as active random field (ARF) [1]. Gibbs Sampling is a widely used tool for computing maximum a posteriori (MAP) estimation (inference), and it is applied in various domains. There are lots of research going on, either on training MRF priors or on speeding up the inference. Some recent researches combine Bayesian inference and convolutional neural network (CNN).

3. Methods

Given a noisy image X, where $X(i, j)$ (also denoted as $x_{i,j}$) is the pixel value at row i and column j. Assume the original image is Y. Denoising can be treated as a probabilistic inference, where we perform maximum a posteriori (MAP) estimation by maximizing the a posteriori distribution $p(y|x)$. By Bayes Theorem,

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}.$$

By taking logarithm on both sides, we get

$$\log p(y|x) = \log p(x|y) + \log p(y) - \log p(x).$$

X is given, so MAP estimation corresponds to minimizing

$$-\log p(x|y) - \log p(y).$$

3.1. Markov Random Field

A Markov Random Field (MRF) is a graph $G = (V, E)$, where $V = v_1, v_2, ..., v_N$ is the set of nodes, each of which is associated with a random variable v_i. The neighborhood of node v_i, denoted $N(v_i)$, is the set of nodes to which v_i is adjacent, i.e., $v_j \in N(v_i)$ and $(v_i, v_j) \in E$. In a MRF, given $N(v_i)$, v_i is independent on the rest of the nodes. Therefore, $N(v_i)$ is often called the Markov blanket of node v_i.

A classic structure used in image denoising domain is the pairwise MRF, shown in Figure 1. The yellow nodes (original image Y) are what we want to find, but we only have information about the green nodes (noisy image X). Each node y is only connected with its corresponding output x and four direct neighbors (up, down, left, right). Therefore, given a pixel y its 5 neighbors, we can determine the probability distribution of y without looking into other pixels.

3.2. Gibbs Sampling

In its basic version, Gibbs sampling is a special case of the MetropolisHastings algorithm. It is a Markov chain Monte Carlo (MCMC) algorithm that samples each random variable of a graphical model, one at a time. The point of Gibbs sampling is that given a multivariate distribution it is simpler to sample from a conditional distribution than to marginalize by integrating over a joint distribution. In our case, we sample one value of a single pixel $y_{i,j}$ at a time, while keeping everything else fixed. Assume the input image has a size of $M \times N$. The algorithm is as shown in Algorithm 1.

3.2.1 Gibbs Sampling for binary images

As a warn-up and small test on the efficiency of Gibbs Sampling and pairwise MRF, I implemented an algorithm for binary images. Figure 2 shows the work. The original image contains a clean formula. Then, I generated an noisy image such that each pixel from the original has a 20% chance of been flipped. I set the sampling probability of each pixel using the Ising model. After only a few iterations, I got the output which recovered most part of the original image with an error rate of 0.8%.

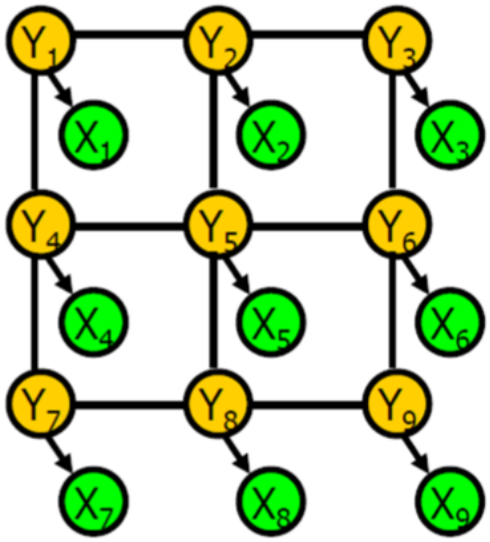

Figure 1. Pairwise MRF.

Initialize starting values of $y_{i,j}$ for $i = 1, ..., M$ and $j = 1, ..., N$;
while *not at convergence* **do**
  Pick an order of the $M \times N$ variables;
  for *each variable* $y_{i,j}$ **do**
    Sample $y_{i,j}$ based on $P(y_{i,j}|N(y_{i,j}))$;
    Update $y_{i,j}$ to Y;
  end
end

Algorithm 1: Gibbs Sampling

3.3. Energy

The distribution over random variables can be expressed in terms of an energy function E. Here we define the energy E for variable $y_{i,j}$ as a sum of loss (L), sparse gradient prior (R) and window-wise products (W)

$$E(y_{i,j}|X) = E(y_{i,j}|N(y_{i,j}))$$

$$= L(y_{i,j}|N(y_{i,j})) + \lambda_r R(y_{i,j}|N(y_{i,j})) - \lambda_w W(y_{i,j}|N(y_{i,j}))$$

The probability distributions of $y_{i,j}$ is defined as

$$p(y_{i,j}|N(y_{i,j})) = \frac{1}{Z} exp(-E(y_{i,j}|X)),$$

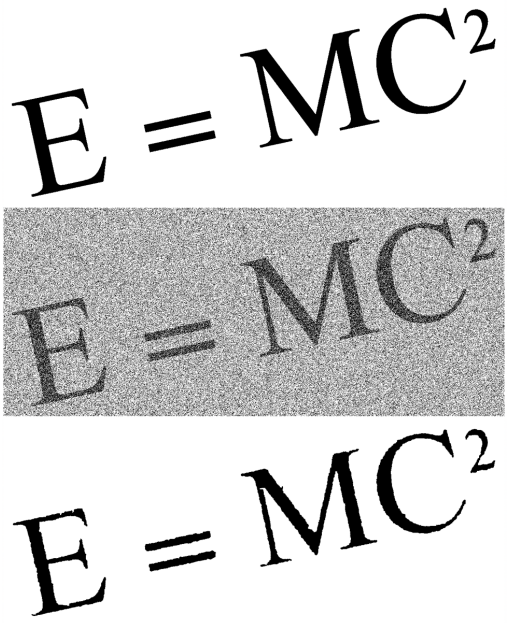

Figure 2. Using Gibbs Sampling and Pairwise MRF for binary image denoising. From top to bottom: original image, noisy image with a flip rate of 20%, de-noised image with an error of 0.8%

where Z is the normalization factor. Therefore, higher energy means lower chance of getting sampled, the way we compute the energy depends on MRF.

3.3.1 Loss

Loss is defined as the closeness of $y_{i,j}$ to its neighbors and corresponding $x_{i,j}$. Therefore, the closer they are, the smaller the loss. We can define loss as the sum of the norm of distances:

$$L(y_{i,j}|X) = \sum_{z \in N(y_{i,j}) \setminus x_{i,j}} ||z - y_{i,j}||_{n_1}^{n_1} + \lambda ||x_{i,j} - y_{i,j}||_{n_1}^{n_1}.$$

The first term penalizes the difference between $y_{i,j}$ and its neighbors in y, and the second term penalizes the difference between $y_{i,j}$ and the noisy pixel $x_{i,j}$. I found that $\lambda = 1$ is a good setting, and also helps with vectorization. The problem with L-2 norm is that it could be largely effected by outliers, but L-1 could create unnecessary patches.

Lorentzian function is robust to outliers, and does not generate as many patches as L-1 norm. It is written as:

$$\rho(z, \sigma) = log(1 + \frac{1}{2}(\frac{z}{\sigma})^2).$$

By plugging this into the loss function, we get:

$$L(y_{i,j}|X) = \sum_{z \in N(y_{i,j}) \setminus x_{i,j}} \rho(z - y_{i,j}, \sigma) + \rho(x_{i,j} - y_{i,j}, \sigma).$$

σ is a hyper parameter here, it controls the bandwidth.

3.3.2 Sparse gradient prior

We can also add our assumptions about images as a prior term here. I assumed the image has sparse gradient. Therefore, we also penalize for large gradients. The prior term is:

$$R(y_{i,j}|X) = \sum_{z \in N(y_{i,j}) \setminus x_{i,j}} ||\frac{z - y_{i,j}}{z}||_{n_2}^{n_2}.$$

Where here I approximated the gradient between $y_{i,j}$ and z as the ratio of their difference to value of z, because this makes vectorization easy. Again, there is a trade-off between L-1 and L-2 norms.

3.3.3 Window-wise product

Inspired from the Non-local Means (NL-means) method [3] and Fields of Experts [7], I thought it could be helpful to derive common patterns within an image, instead of training on some dataset. The advantage of doing this is it saves time, and could be scene-specific. Although the disadvantage is also obvious: could be biased. If we can find 'patterns' within a $m \times n$ window, then dot product these patterns with the window the Gibbs sampler is currently looking at could yield some useful information. The most straight forward pattern is just the mean: go through the image with a $m \times n$ sliding window and then average the sum of pixel values that the window has seen. This turned out to be not helpful, since the mean tend to be plain. The second thing I tried was PCA: taking out the first s principle components and their corresponding explained variance. I denote components as filters f, the variance explained value as α, and the current window as w. Therefore, the third term in our energy function becomes:

$$W(y_{i,j}|X) = \sum_{k=1}^{k=s} \alpha_k |f_k \cdot w|$$

4. Results and analysis

PSNR and MSE are the two major criteria, results having higher PSNR generally will have lower MSE. Computation time is also considered. I used Gaussian smoothing as the baseline model, and Non-local Means as the state of art (in general, BM3D performs better, but since it is not a free package in OpenCV, I used NLM). The original image is shown in Figure 3, every color of a pixel reads in as an integer ranging from 0 to 255. After applying a Gaussian noise with standard deviation of 100 and clipping the out of boundary values, the noisy image is shown in Figure 4. It has a PSNR of 10.26 dB. Gaussian smoothing can get to a PSNR of 18.56 dB, and NL-means gives 26.99 dB, as

Figure 3. Original image

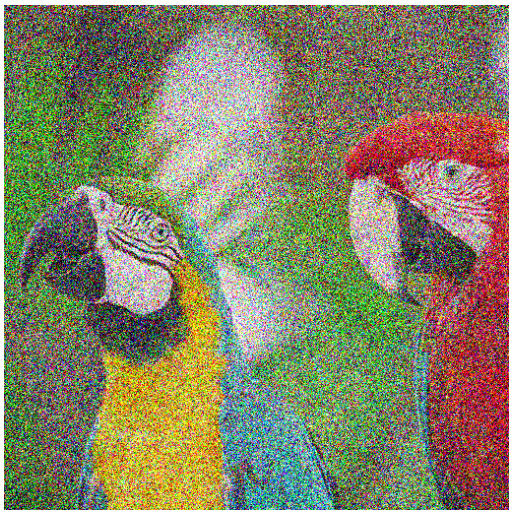

Figure 4. Noisy image (X), σ = 100, PSNR = 10.26 dB

shown in Figure 5. Though the PSNR of non-local means method is very impressive, it didn't remove the noise: the high PSNR came from the low MSE, because the de-noised image is very close to the original one.

4.1. MRF with different neighbor size

I tested the influence of the number of neighbors. I expanded the Markov blanket of pixel $y_{i,j}$ such that it is also connected to its diagonal neighbors, i.e., $y_{i-1,j-1}$, $y_{i-1,j+1}$, $y_{i+1,j-1}$ and $y_{i+1,j+1}$. I call this a 3×3 window. I also tested the 5×5 window. The finding is that in general, 3×3 window is better and more robust than simple pairwise and 5×5 window for all types of norms. At very high

Figure 5. After de-noised by default OpenCV non-local-means function, PSNR = 26.99 dB. Though the PSNR is very impressive, it didn't remove the noise: the high PSNR came from the low MSE, because the de-noised one is very close to the original.

noise level (e.g., $\sigma = 100$ for a 0 to 255 value range), 5×5 gives higher PSNR value, but it is not visually good, since it creates too many 'patches'. I decided to use 3×3 window for the rest testings.

4.2. MRF with different norms

The results vary a lot due to loss functions, and I tested my image with energy function only including the loss and prior terms. Figure 6 shows the result after using L-1 norm for both loss and sparse-gradient prior. It has a PSNR of 23.34 dB. Figure 7 shows the result after using L-2 norm for both terms, and it has a PSNR of 21.08 dB. Although L-1 norm gives a higher PSNR, it is not visually good.

The Lorentzian function is robust to outliers, I tried using that as the loss but keep L-2 norm as the sparse gradient function. And it gives a much better result, as shown in Figure 8. It has a PSNR of 23.85 dB.

4.3. MRF with the idea of window-wise pattern

After choosing Lorentzian function for loss and L-2 norm for prior, I added another term to the energy function: weighted dot products of window and first few principal components. Not surprisingly, this helped improve PSNR. As it turned out, the larger the sum of dot products, the lower energy that node should has: it does not want a pixel to be very off from the mean. Figure 9 shows this ensemble result, and it has a PSNR of 24.46 dB. If more L-1 loss were added, the PSNR would further increase, but that is not visually good.

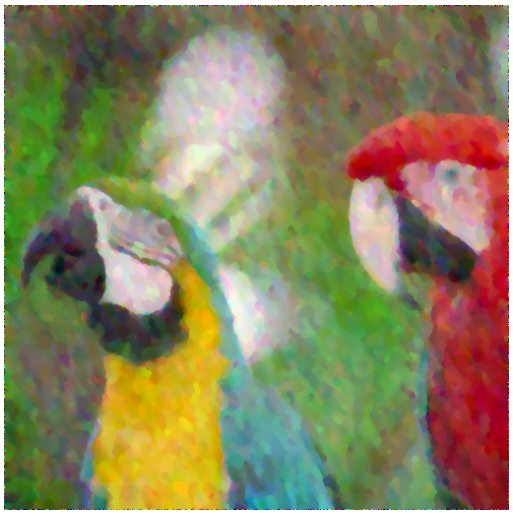

Figure 6. Using L-1 norm for loss and prior, PSNR = 23.34 dB.

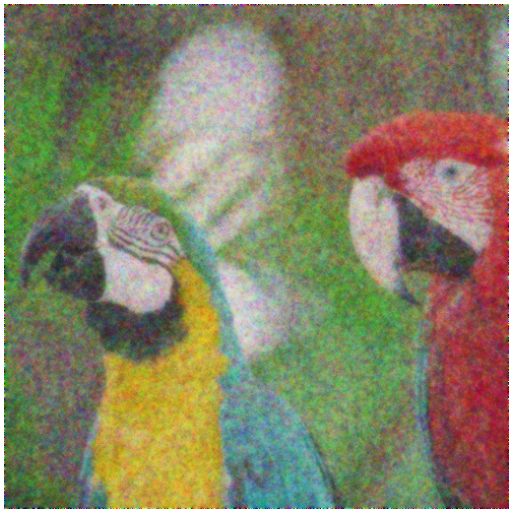

Figure 7. Using L-2 norm for loss and prior, PSNR = 21.08 dB. L-2 is visually better, although it has a much lower PSNR than L-1.

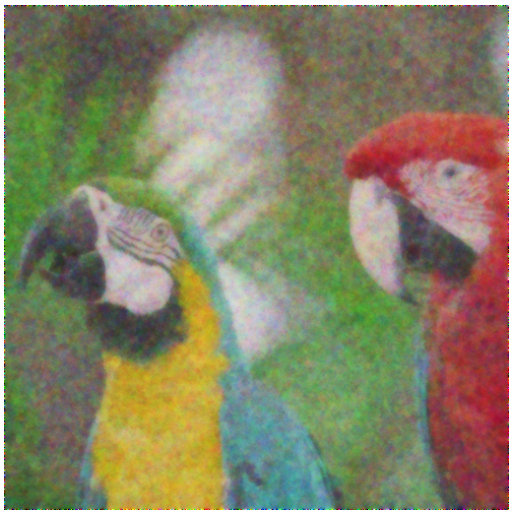

Figure 8. Using Lorentzian function for loss and L-2 for prior, PSNR = 23.85 dB.

Figure 9. Ensemble method, PSNR = 24.46 dB. If more L-1 loss were added, the PSNR would further increase, but that won't be visually good.

4.4. Analysis and discussion

Another example of Gibbs Sampling is shown from Figure 10 to Figure 15. It used the same image but with a lower noise level: $\sigma = 50$. Gibbs sampling can get a PSNR close to non-local means easily, and it is much superior than the vanilla Gaussian Smoothing method. However, it it is also time consuming: all images were generated from 8 burn-ins and 4 samplings, and it runs on the order of minutes (since my code is not optimized, it has a room for improvement). Another drawback is that I need to tune parameter to balance terms in the energy function. But NL-means also require parameter tuning to yield good results. In OpenCV, Gaussian smoothing runs in second, and NL-means in tens seconds.

I like Gibbs Sampling because it is guaranteed to converge and leaves flexibility to design the MRF. The sampling process is also simple, since you only need to look at pixels in the current pixel's Markov blanket, a type of divide-and-conquer. In the future, I might apply various loss functions such as Huber. I might also try filters other than PCA.

Figure 10. Noisy image (X), σ = 50, PSNR = 14.78 dB

Figure 11. After de-noised by default OpenCV non-local-means function, PSNR = 27.93 dB.

Figure 12. Using L-1 norm for loss and prior, PSNR = 25.94 dB.

Figure 13. Using L-2 norm for loss and prior, PSNR = 25.50 dB. L-2 is visually better.

References

[1] A. Barbu. Training an active random field for real-time image denoising. *IEEE Transactions on Image Processing*, 18(11):2451–2462, 2009.

[2] J. F. Brinkley and C. Rosse. Imaging informatics and the human brain project: the role of structure. *Yearbook of Medical Informatics*, 11(01):131–148, 2002.

[3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[4] W. R. Gilks, N. Best, and K. Tan. Adaptive rejection metropolis sampling within gibbs sampling. *Applied Statistics*, pages 455–472, 1995.

[5] S. Z. Li. Markov random field models in computer vision. In *European conference on computer vision*, pages 361–370. Springer, 1994.

[6] J. J. McAuley, T. S. Caetano, A. J. Smola, and M. O. Franz. Learning high-order mrf priors of color images. In *Proceedings of the 23rd international conference on Machine learning*, pages 617–624. ACM, 2006.

[7] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 860–867. IEEE, 2005.

[8] U. Schmidt. Learning and evaluating markov random fields for natural images. *Masters thesis*, February 2010.

Figure 14. Using Lorentzian function for loss and L-2 for prior, PSNR = 25.76 dB.

Figure 15. Ensemble method, PSNR = 26.30 dB, comparable to NL-means. If more L-1 loss were added, the PSNR would further increase, but that won't be visually good.