Application of Variational Inference in Boltzmann Machines for Denoising Binary Images

Zhongwei Huang

Xiaolong Shen

Kewei Zhou

Contents

Objective of the Study 3
Data 3
Introduction to Boltzmann Machines 3
Introduction to Denoising Images with Boltzmann Machines 3
Introduction to Variational Inference 4
Derivation of Approximating the Posterior for A Boltzmann Machine 5
Algorithm 6
Main Findings 7
Parameters Tuning 7
Reconstructing Images 8
Appendix ..... **Error! Bookmark not defined.**
Code ..... **Error! Bookmark not defined.**

Objective of the Study

In this project, we used Boltzmann Machine to denoise the image data. When using Boltzmann Machine, the posterior of Boltzmann Machine is hard to calculate, which we will use variation inference method to approximate.

Data

The data we used can be found from MNIST database of hand written digits, which can be downloaded from http://yann.lecun.com/exdb/mnist/. There are 4 compressed files in the link, the one we used is called "`train-images-idx3-ubyte.gz`", which is the training set images, and we only used the first 500 images for our task. The data contains the pixel value of images. Each image has 28*28 pixels, and the images are all digit numbers from 0 to 9. There is no noise in the original data. To perform the denoise task, we added noise to the data by flipping 2% of the pixels values.

Introduction to Boltzmann Machines

Boltzmann machine is a distribution model for a set of binary random variables. Assume we have N binary random variables U_i, which take the values 1 or -1. The values of these random variables are not observed (the true values of the pixels). Also, we will assume that some random variables are coupled, so they are not independent. Write $\mathcal{N}(i)$ for the set of random variables whose values are coupled to that of i. The joint probability model is

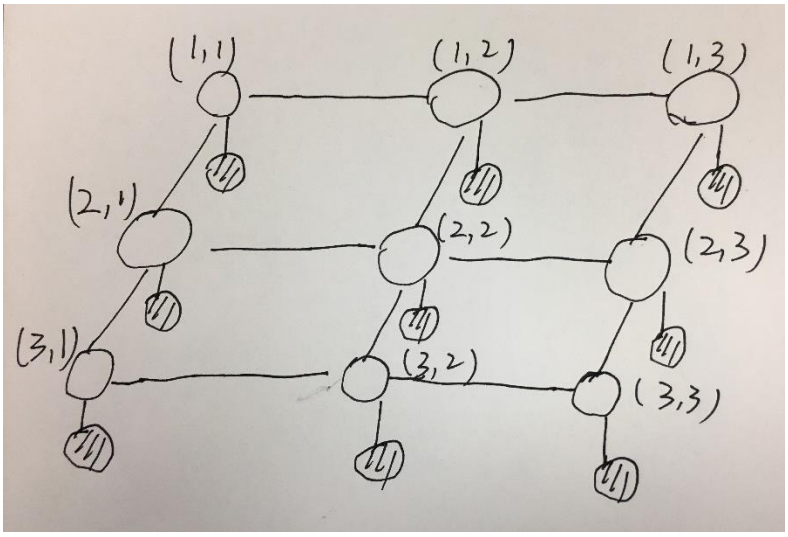
$$logP(U|\theta) = [\sum_{i} \sum_{j \in \mathcal{N}(i)} \theta_{ij} U_i U_j] - logZ(\theta)$$

Now $U_i U_j$ is 1 when U_i and U_j agree, and -1 otherwise. The θ_{ij}'s measure the weights of coupling strengths; notice if $\theta_{ij} > 0$, the model generally prefers U_i and U_j to agree, and if $\theta_{ij} < 0$, the model prefers they disagree. $Z(\theta)$ ensures that the model normalizes to 1, so that

$$Z(\theta) = \sum_{all\ values\ of\ U} exp(\sum_{i} \sum_{j \in \mathcal{N}(i)} \theta_{ij} U_i U_j)$$

Introduction to Denoising Images with Boltzmann Machines

Here is a simple model for a binary image that has been corrupted by noise. At each pixel, we observe the corrupted value, which is binary, while hidden from us are the true values of each pixel. The true value at each pixel is affected by the true value at each of its neighbors. By applying a Boltzmann machine, we can denoise the image. We split the U into two groups. One group represents the observed value at each pixel, X_i, and the other represents the hidden value at each pixel, H_i.



As shown above, the shaded circle represents the observed pixels, and the empty circle represents the hidden value of pixels. Each observation is either 1 or -1. We assume that each pixel is coupled with its neighbors. For example, the neighbors of pixel (1,1) are (1,2) and (2,1), and the neighbors of pixel (2,2) are (2,1), (2,3), (1,2) and (3,2). We also assume that H_i and X_i are coupled. The joint probability model is

$$logP(U|\theta) = logP(H, X|\theta) = \sum_i \sum_{j \in \mathcal{N}(i) \cap H} \theta_{ij} H_i H_j + \sum_i \sum_{j \in \mathcal{N}(i) \cap X} \theta'_{ij} H_i X_j + K$$

where K doesn't depend on any H_i.

Introduction to Variational Inference

Variational inference (VI) is used to approximate difficult-to-compute probability densities. The idea behind VI is to first posit a family of densities and then to find a member of that family which is close to the target density. Closeness is measured by Kullback-Leibler divergence.

First, we set up the general problem. Consider a joint density of latent variables $H = H_{1:m}$ and observations $X = X_{1:n}$,

$$P(H, X) = P(H)P(X|H)$$

In Bayesian models, the latent variables help govern the distribution of the data. A Bayesian model draws the latent variables from a prior density $P(H)$ and then relates them to the observations through the likelihood $P(X|H)$. In complex Bayesian models, computation of the posterior $P(H|X)$ often requires approximate inference. In VI, we first posit a family of approximate densities Q. This is a set of densities over the latent variables. Then, we try to find the member of that family that minimizes the KL divergence to the exact posterior,

$$Q^*(H) = arg\,min_{Q(H) \in Q}\, KL(Q(H)||P(H|X))$$

which is equal to

$$Q^*(H) = arg\,max_{Q(H)\in Q}\, ELBO(Q(H)) = arg\,max_{Q(H)\in Q}\, E_Q[logP(H,X)] - E_Q[logQ(H)]$$

In addition to the objective function ELBO, we now describe a variational family Q to complete the specification of the optimization problem. We focus on the mean-field variational family, where the latent variables are mutually independent and each governed by a distinct factor in the variational density. A generic member of the mean-field variational family is

$$Q(H) = \prod_{i=1}^{m} q_i\,(H_i)$$

Each latent variable H_i is governed by its own variational factor, the density $q_i(H_i)$.

Using the ELBO and the mean-field family, and fixing the other variational factors $q_j(H_j)$, $j \neq i$, the optimal $q_i(H_i)$ is proportional to the exponentiated expected log of the joint,

$$q_i^*(Hi) \propto exp(E_{Q_{-i}}[logP(H_i, H_{-i}, X)])$$

where Q_{-i} is the distribution obtained by omitting $q_i(H_i)$ from the product. Therefore, to maximize ELBO, we iteratively update $q_i(H_i)$ given all the other $q_j(H_j)$, $j \neq i$ until ELBO converges.

Derivation of Approximating the Posterior for A Boltzmann Machine

We use the mean-field family to approximate the posterior, so $Q(H) = q_1(H_1)q_2(H_2)\ldots q_N(H_N)$. Because H_i can only take two possible values, which are 1 and -1,

$$q_i(H_i) = \pi_i^{(1+H_i)/2}(1-\pi_i)^{(1-H_i)/2},\ where\ \pi_i = p(H_i = 1)$$

We wish to maximize the ELBO, which is

$$ELBO(Q(H)) = E_Q[logP(H,X)] - E_Q[logQ(H)]$$

We first look at the $E_Q[logQ(H)]$ term. We have

$$\begin{aligned} E_Q[logQ(H)] &= E_{q_1(H_1)q_2(H_2)\ldots q_N(H_N)}[logq_1(H_1)+\ldots+logq_N(H_N)] \\ &= E_{q_1(H_1)}[logq_1(H_1)]+\ldots+E_{q_N(H_N)}[logq_N(H_N)] \end{aligned}$$

Now we need to deal with $E_Q[logP(H,X)]$. We have

$$
\begin{aligned}
E_Q[logP(H,X)] &= E_Q[\sum_i \sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\, H_i H_j + \sum_i \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\, H_i X_j + K] \\
&= E_Q[\sum_i \sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\, H_i H_j] + E_Q[\sum_i \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\, H_i X_j] + K \\
&= \sum_i (\sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\, E_{q_i(H_i)q_j(H_j)}[H_i H_j] + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\, X_j E_{q_i(H_i)}[H_i]) + K
\end{aligned}
$$

where K doesn't depend on any H_i.

Because $q_i^*(Hi) \propto exp(E_{Q_{-i}}[logP(H_i, H_{-i}, X)])$, we need to compute $E_{Q_{-i}}[logP(H_i, H_{-i}, X)]$ to update $q_i(H_i)$. If $H_i = -1$, we have

$$
\begin{aligned}
E_{Q_{-i}}[logP(-1, H_{-i}, X)] &= E_{Q_{-i}}[\sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\, H_i H_j + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\, H_i X_j + terms\ not\ containing\ H_i] \\
&= \sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(-1) E_{Q_{-i}}[H_j] + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(-1) X_j + K_i \\
&= \sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(-1)((\pi_j)(1) + (1-\pi_j)(-1)) + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(-1) X_j + K_i \\
&= \sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(-1)(2\pi_j - 1) + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(-1) X_j + K_i
\end{aligned}
$$

where K_i doesn't depend on H_i. Hence, we have

$$
q_j(-1) \propto exp(\sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(-1)(2\pi_j - 1) + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(-1) X_j)
$$

Similarly, we have

$$
q_j(1) \propto exp(\sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(1)(2\pi_j - 1) + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(1) X_j)
$$

Therefore, we have

$$
= \frac{exp(\sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(1)(2\pi_j - 1) + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(1) X_j)}{exp(\sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(1)(2\pi_j - 1) + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(1) X_j) + exp(\sum_{j\in\mathcal{N}(i)\cap H} \theta_{ij}\,(-1)(2\pi_j - 1) + \sum_{j\in\mathcal{N}(i)\cap X} \theta'_{ij}\,(-1) X_j)}
$$

Algorithm

Now we can describe the algorithm for denoising a corrupted image.

Input: An image corrupted by noises, $\theta_{ij} = 0.2$, $\theta'_{ij} = 0.2$

Output: Variational factors $q_i(H_i = 1),\ i = 1,2,\ldots,N$, where N is the number of pixels in the image

Initialize: $q_i(Hi = 1) = 0.5,\ i = 1,2,\ldots,N$

while the ELBO has not converged **do**

for $i \in 1,..,N$ **do**

Set $q_i(H_i = 1) = \frac{exp(\sum_{j\in\mathcal{N}(i)\cap H}\theta_{ij}(1)(2\pi_j-1)+\sum_{j\in\mathcal{N}(i)\cap X}\theta'_{ij}(1)X_j)}{\sum_{H_i}exp(\sum_{j\in\mathcal{N}(i)\cap H}\theta_{ij}(H_i)(2\pi_j-1)+\sum_{j\in\mathcal{N}(i)\cap X}\theta'_{ij}(H_i)X_j)}$

end

Compute $ELBO(Q) = E_Q[logP(H,X)] - E_Q[logQ(H)]$

end

return $q_i(H_i)$

When computing the ELBO, we can simply omit K, because we just need to compare a new ELBO with the previous ELBO to check whether the ELBO has converged.

Main Findings

Parameters Tuning

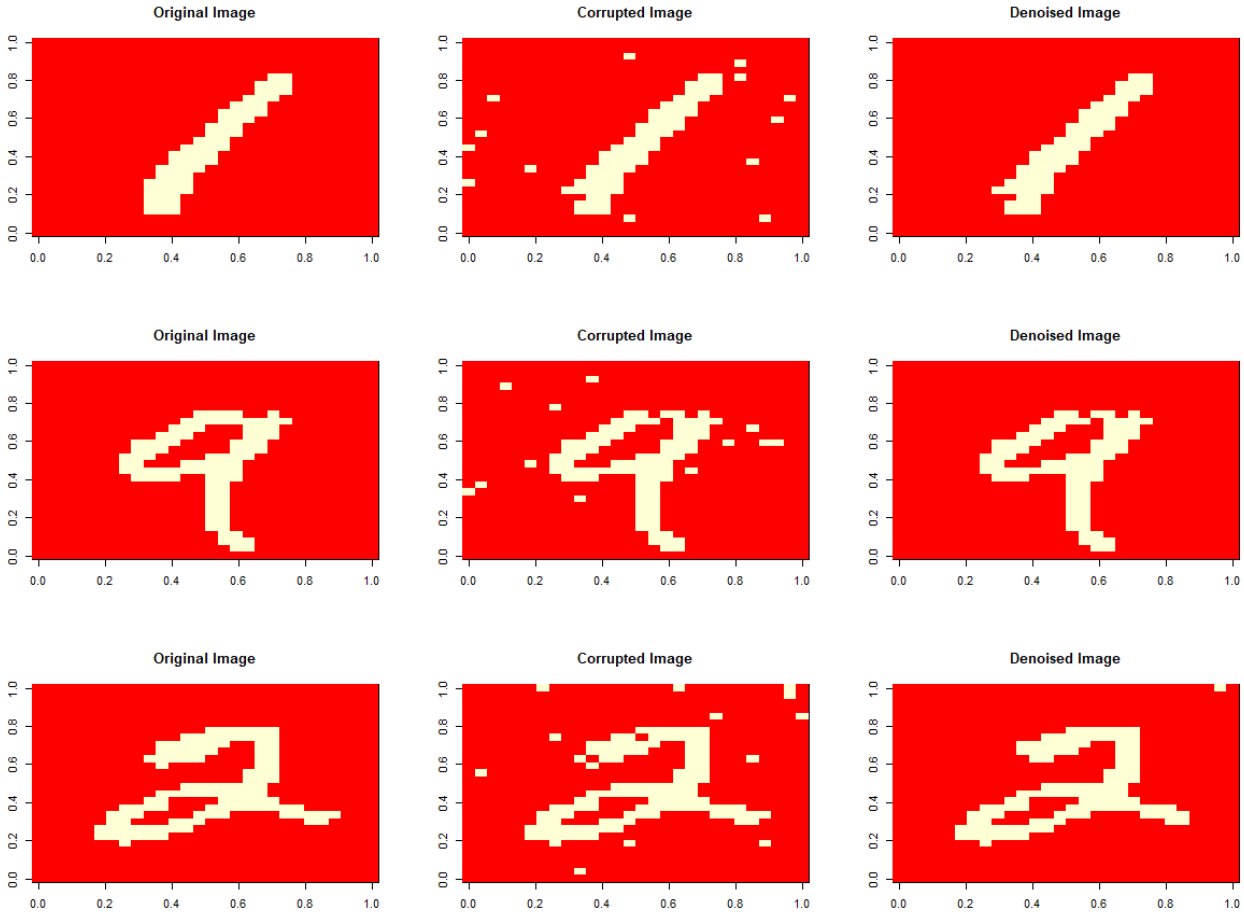
To construct the best denoising model, we tried different value of θ_{ij}, θ'_{ij}. We tried different value of (-1, 0, 0.2, 1, 2) for θ_{ij}, and θ'_{ij}, and compared the accuracy of reconstructed images. The accuracy here is calculated by evaluating how many pixels of the denoised image is the same to the original image. The corresponding value and accuracy of the denoised images are as follow,

	theta	theta_	Accuracy
[1,]	-1.0	-1.0	49.129082
[2,]	-1.0	0.0	87.265306
[3,]	-1.0	0.2	50.217602
[4,]	-1.0	1.0	50.870918
[5,]	-1.0	2.0	57.191327
[6,]	0.0	-1.0	2.040816
[7,]	0.0	0.0	87.265306
[8,]	0.0	0.2	97.959184
[9,]	0.0	1.0	97.959184
[10,]	0.0	2.0	97.959184
[11,]	0.2	-1.0	2.040816
[12,]	0.2	0.0	87.265306
[13,]	0.2	0.2	99.436480
[14,]	0.2	1.0	97.959184
[15,]	0.2	2.0	97.959184
[16,]	1.0	-1.0	3.784184
[17,]	1.0	0.0	87.265306
[18,]	1.0	0.2	87.265306
[19,]	1.0	1.0	96.215816
[20,]	1.0	2.0	99.397959
[21,]	2.0	-1.0	12.737755
[22,]	2.0	0.0	87.265306
[23,]	2.0	0.2	86.542092
[24,]	2.0	1.0	87.262245
[25,]	2.0	2.0	87.265306

According to the result, the best combination of parameters are θ_{ij}=0.2, θ'_{ij}=0.2, with the accuracy of 99.43%.

Reconstructing Images

Using θ_{ij}=0.2, θ'_{ij}=0.2, we can reconstruct the corrupted images now. The images below are selected comparison of original images, corrupted images, and denoised images.



We can see that the denoised image is very similar to the original image, which suggest that our algorithm for denoising is robust.