Contrastive Divergence Learning with Chained Belief Propagation

Fan Ding DING274@PURDUE.EDU

Yexiang Xue YEXIANG@PURDUE.EDU

Department of Computer Science, Purdue University, USA

Abstract

Contrastive Divergence (CD) is an important maximum-likelihood learning approach for probabilistic graphical models. CD maximizes the difference in likelihood between the observed data and those sampled from the current model distribution using Markov Chain Monte Carlo (MCMC). Nevertheless, the overall performance of CD is hampered by the slow mixing rate of MCMC in the presence of combinatorial constraints. A competing approach BP-CD replaces MCMC with Belief Propagation (BP). However, their samples are generated from a mean-field approximation, which may be far away from the true distribution. Here we propose contrastive divergence learning with chained belief propagation (BPChain-CD). To generate one sample in CD, we fix one variable at a time based on the marginal distribution computed by BP conditioned on previous variables. We analyze BPChain-CD both theoretically and experimentally. We show that BPChain-CD learns better models compared with BP-CD and CD on a range of maximum-likelihood learning experiments.

Keywords: Contrastive Divergence; Belief Propagation; Maximum Likelihood Learning.

1. Introduction

The Contrastive Divergence (CD) algorithm has achieved notable success in training energy-based models including Markov random fields (MRF) and Restricted Boltzmann Machines (RBM) (Hinton, 2002; Carreira-Perpinan and Hinton, 2005; Bengio and Delalleau, 2009; Sutskever and Tieleman, 2010; Ceylan and Gutmann, 2018; Jiang et al., 2018; Ruiz and Titsias, 2019) and played a key role in the emergence of deep learning (Goodfellow et al., 2014; Salakhutdinov, 2015). The idea is to transform the problem of computing the intractable partition function into approximating the expectation of the gradient of the log-partition function, where a bunch of sampling methods can be taken advantage of to approximate the expectation. The approximate gradient is computationally-cheap. Therefore, the quality of samplers is of great importance.

Traditional CD used a k-step (CD-k) Markov Chain Monte Carlo (MCMC) sampling methods (Plummer et al., 2006; Andrieu et al., 2003; Hinton, 2012) to attack this problem. To speed up the convergence of Markov chain, an important variant of CD-k called persistent CD (PCD) (Tieleman, 2008; Tieleman and Hinton, 2009) used a persistent Markov chain during learning to provide a better approximation to the target distribution than the limited step chain in CD-k. Further work also employed approximate inference methods, such as mean-field (MF) and BP as inference routines in learning Contrastive Divergence (Yedidia et al., 2001; Murphy et al., 2013; Hershey et al., 2014). Recently an efficient implementation of BP algorithms (BP-CD) has been proposed (Ping and Ihler, 2017) to deal with MRF and RBM on a large scale.

However, there are fundamental limitations of those approaches. Both of CD-k and PCD do suffer from learning graphical models with multi-modes. CD methods stagnate when exploring the peaks of multi-modal distributions in a generative setting due to the low acceptance rate to move across peaks. Although approximate inference methods are efficient, MF is conceptually problem-

atic in the sense that it effectively maximizes an upper bound of the log-likelihood in learning. In addition, MF uses a unimodal proposal to approximate the multi-modal distribution, which may lead to unsatisfactory results. Loopy BP usually provides a better approximation of marginals than MF (Li and Zemel, 2014; Domke, 2013). However, the fundamental problem still exists when we sample with BP. Despite the better marginals it provides, its samples are generated from a mean-field approximation, which may be far away from the true model distribution.

In this work, we propose to embed a chain of Belief Propagation procedures into Contrastive Divergence (BPChain-CD). Different from previous methods, this BPChain sampling schedule could efficiently solve the mean-field problem arisen by BP and the slow mixing rate problem of MCMC. Instead of sampling in a mean-field manner, BPChain generates samples sequentially according to a conditional probability chain. More specifically, to generate one sample in each gradient descent iteration of CD, we fix one variable at a time based on the marginal distribution computed by BP conditioned on previously generated variables. Because of this different sampling approach, BPChain-CD generates samples more likely from the joint model distribution, rather than each marginal when BP gives a sufficiently good approximate marginal in a loopy graph. Therefore, BPChain-CD has the ability to guide the gradient descent in more correct direction, making the algorithm converge faster and fitting training data more precisely.

Empirical experiments demonstrate that BPChain-CD learns better models compared to CD and BP-CD in a series of maximum-likelihood learning experiments. We demonstrate that, under the multi-modal setting of Markov random fields, learning MRF models with BPChain-CD can provide much higher average likelihood than the state-of-the-art CD methods. We also show in a structured sequence generation task that our algorithm learns the most suitable model of some given sequences, while traditional CD with Gibbs sampling learns badly and BP-CD is heavily biased.

The contribution of this paper can be summarized as follows: (1) We addressed potential problems of MCMC and BP as the sampler in Contrastive Divergence learning. (2) We proposed BPChain, a conditional probability chain of BP, to sample from multi-modal distributions where each dimension is correlated with each other. (3) We formulated BPChain-CD by embedding BPChain in the CD framework, exhibiting a superior learning ability compared to both CD and BP-CD. (4) Experimental results on discrete MRF and structured sequences generation showed superior performance of our method.

2. Preliminaries

In this section, we review some results on probabilistic graphical models and Contrastive Divergence for Maximum-Likelihood (ML) Learning (Carreira-Perpinan and Hinton, 2005).

2.1 Markov random field

We consider a graphical model specified as a factor graph with $N = |V|$ discrete random variables $X_i \in \mathcal{X}_i, i \in V$ where $\mathcal{X}_i = \{0, 1\}$. The global random vector $X = [X_1, X_2, \ldots, X_N]$ takes value in the cartesian product $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$. We consider a function over $X \in \mathcal{X}$ as follows:

$$f(X; \Theta) = \prod_{\alpha \in \mathcal{I}} \phi_\alpha(\{X\}_\alpha, \Theta_\alpha)$$

which factors into potentials $\phi_\alpha : \{X\}_\alpha \to \mathbb{R}^+$, where $\mathcal{I}$ is the set of all the cliques of the graph, $\{X\}_\alpha \subseteq V$ is a subset of variables that the factor ϕ_α depends on, and Θ is a vector of model

parameters. We consider a normalized distribution $p(X;\Theta) = \frac{1}{Z(\Theta)} f(X;\Theta)$ where $Z(\Theta)$, the normalization constant, also known as the partition function, is defined as $Z(\Theta) = \sum_X f(X;\Theta)$. Notice however that computing $Z(\Theta)$ is normally intractable in practice.

2.2 Contrastive Divergence Learning

In the context of graphical models learning, we want to fit a set of given data points $\{x_k\}_{k=1}^K$, $x_k \in \{0,1\}^N$, using a graphical model $p(X;\Theta)$ with unknown model parameters Θ. For convenience, in this paper we use index k to denote the k-th sample, and we use index i to denote the i-th dimension for both random vector X and sample x. Taking those given data points $\{x_k\}_{k=1}^K$ as the training set, we learn our model parameters by maximizing the probability of the training set $\max_\Theta p(X;\Theta)$, which is equivalent to minimize the negative log of $p(X;\Theta)$, denoted as $E(X;\Theta)$

$$\min_\Theta E(X;\Theta) = \min_\Theta \Big(\log Z(\Theta) - \frac{1}{K} \sum_{k=1}^K \log f(x_k;\Theta) \Big)$$

Contrastive Divergence (CD) learning (Hinton, 2002) has been successfully applied to learn $E(X;\Theta)$ by avoiding directly computing the intractable $Z(\Theta)$. In each iteration step of gradient descent, CD estimates the gradient of $E(X;\Theta)$. Given the partial derivative

$$\frac{\partial E(X;\Theta)}{\partial \Theta} = \frac{\partial \log Z(\Theta)}{\partial \Theta} - \Big\langle \frac{\partial \log f(X;\Theta)}{\partial \Theta} \Big\rangle_X$$

where $\langle \cdot \rangle_X$ is the expectation of $\cdot$ given the data distribution of X. Here the first term can be substituted as $\frac{\partial \log Z(\Theta)}{\partial \Theta} = \Big\langle \frac{\partial \log f(X;\Theta)}{\partial \Theta} \Big\rangle_{p(X;\Theta)}$. Although this expectation is generally intractable, it could be numerically approximated by drawing samples from the proposed distribution $p(X;\Theta)$. Sampling from $p(X;\Theta)$ requires knowledge of the partition function which is unknown; therefore, sampling techniques such as MCMC use many cycles to transform the original training data $\{x_k\}_{k=1}^K$ into data drawn from the model distribution $p(X;\Theta)$. Using such a sampling scheme, we can take gradient descent to devise an updating rule for the parameters Θ

$$\Theta^{t+1} = \Theta^t + \eta \Big(\Big\langle \frac{\partial \log f(X;\Theta^t)}{\partial \Theta} \Big\rangle_{X^0} - \Big\langle \frac{\partial \log f(X;\Theta^t)}{\partial \Theta} \Big\rangle_{X^p} \Big)$$

where X^0 is the distribution of training set $\{x_k\}_{k=1}^K$, X^p is the surrogate model distribution of samples drawn indirectly from the model distribution, and η is the learning rate.

2.3 Sampling in Contrastive Divergence

MCMC (Andrieu et al., 2003) is widely used to transform samples from the training set to those from the model distribution. Additionally, Belief Propagation (Yedidia et al., 2001; Murphy et al., 2013) could also help sample from the model distribution by sampling from marginal distributions of each dimension X_i individually. Here we will briefly introduce these two kinds of algorithms and analyze each sampling procedure's potential problems in the next section.

Gibbs Sampling. MCMC takes advantage of a Markov Chain to sequentially sample from the model distribution. As a special case of MCMC, Gibbs Sampling is widely used in Contrastive Divergence for training discrete probabilistic graphical models. In each MCMC step, Gibbs samples

one dimension based on a conditional marginal distribution. In detail, let $x^t \in \mathbb{R}^N$ denote the current sample, Gibbs sampling proceeds as follows: Firstly, it picks a dimension index $i \in \{1, \cdots, N\}$ either via round-robin or uniformly at random, followed by setting $x^{t+1} = x^t$, for all the index $j \neq i$, i.e, $x_{-i}^{t+1} = x_{-i}^t$, where x_{-i} is all the dimensions of x except the i-th dimension. Then, generate x_i^{t+1} from $p(X_i|X_{-i}^t = x_{-i}^t)$. The acceptance rate of Gibbs sampling is 1 all the time, but the probability of Gibbs sampling to change one dimension from one sample to the next one could be very low in some time, which increases the time of convergence.

Belief Propagation. In Belief Propagation (Yedidia et al., 2001; Murphy et al., 2013) over the distribution $P(X; \Theta)$, each variable X_i can be viewed as a variable node i. In addition, each parameter Θ_α can be viewed as a factor node α. All variable and factor nodes form a bipartite graph. Then, a message from a variable node X_i to a factor node Θ_α is

$$m_{i\to\alpha}(X_i) = \prod_{\alpha' \in N(i)\setminus\{\alpha\}} m_{\alpha'\to i}(X_i)$$

and a message from a factor node Θ_α to a variable node X_i is

$$m_{\alpha\to i}(X_i) = \sum_{X'_i \in \{X\}_\alpha \setminus \{X_i\}} \phi_\alpha(\{X\}_\alpha; \Theta_\alpha) \prod_{i' \in N(\alpha)\setminus\{i\}} m_{i'\to\alpha}(X'_{i'})$$

where $N(i)$ is the set of neighboring factor nodes to i, $N(\alpha)$ is the set of neighboring variable nodes to α, and $\{X\}_\alpha$ is the set of all variables associated with factor node α. We ensure each massage passing process to be normalized. After this iterative procedure finally converging, we can compute the marginal distribution of each variable node X_i as

$$p(X_i) \propto \prod_{\alpha \in N(i)} m_{\alpha\to i}(X_i)$$

Therefore, we can sample each dimension of variable X as $X_i \sim p(X_i)$. Recent work (Ping and Ihler, 2017) embedded BP into Contrastive Divergence to train probabilistic models. They leveraged a compact representation only dependent on matrix product and element-wise operations, which are typically highly optimized in modern high-performance computing architectures.

3. Problems on Gibbs Sampling and BP

Though widely used in practice, Gibbs sampling and Belief Propagation have their own essential problems to some extent. We use a motivating instance in Figure 1 to illustrate the problems. Assuming we have training data X of binary sequences of length 6, which are drawn from the distribution $P(X)$ shown in the left table of Figure 1. The probabilities of drawing 000000 and 111111 are 0.4, and the probabilities of drawing 001100 and 110011 are 0.097, while all of the rest 60 sequences have a probability of 0.0001 to be drawn. We draw a bunch of X from $\{000000, 000001, \ldots, 111111\}$ as training data according to probability $P(X)$. We will show both BP-CD and Gibbs-CD cannot learn the training data well in a reasonable amount of time.

Without the loss of generality, let Gibbs sampling start from the initial sample 000000 and let the model distribution has the same parameters as $P(X)$. Then, the conditional distribution $P(X_i = 1|..., X_{i-1}, X_{i+1}, ...)$ for all $i = 1, 2, ..., 6$ will be $0.0001/(0.0001 + 0.4) = 1/4001$, which means it would in expectation take 4001 steps to change one dimension from 0 to 1. As a

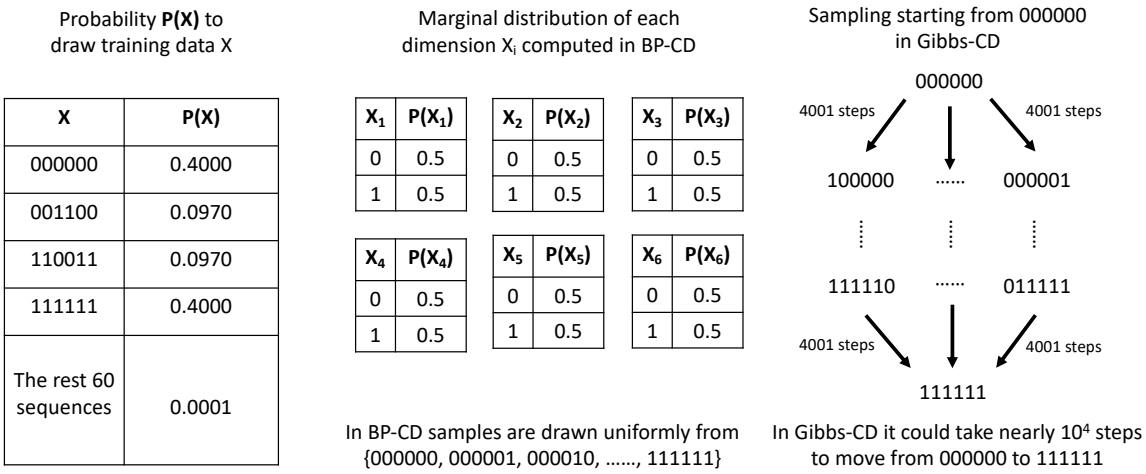


Figure 1: Suppose we have training data X (binary sequence of length 6) obtained by sampling from a distribution $P(X)$ shown in the left table. We show the problems of BP-CD and Gibbs-CD in learning these data. Assuming the model distribution is already well learned, then it should be very close to that of the training data. Since BP-CD computes the marginal distribution for each dimension of X and sample from marginals, the samples are actually drawn uniformly from the set $\{000000, 000001, \ldots, 111111\}$, which is far away from that of training data. In addition, consider Gibbs-CD draws samples starting from 000000, because it leverages each conditional marginal probability to draw samples of each dimension sequentially, in this multi-modal situation it will take nearly 10^4 steps in expectation to move from 000000 to 111111. The biased samples drawn by BP-CD and Gibbs-CD can heavily affect the speed and direction of convergence.

result, moving from 000000 to 111111 would take nearly 10^4 steps. Therefore, samples like 000000 will have a large probability of stagnant or moving not far. As a consequence, the sampling bias of Gibbs-CD will directly affect the speed and direction of gradient descent, leading Θ updated in a different manner. Since samples drawn from Gibbs-CD tend to contain only part of peaks because of the slow mixing rate, they could not make the learning process stop at the right time.

Though sampling by Belief Propagation does not have such a problem, it does suffer from mean-field problem because it treats each dimension of X independent with the others when we use it to sample from marginal distributions of dimension X_i. Still in this motivating instance, we also show that BP-CD fails in learning these training data. Consider we already have a well-learnt model which has the same parameters as $P(X)$, then samples drawn from the model distribution should be similar to the training data, yielding a gradient with respect to Θ close to 0. However, BP-CD first computes all marginals and then samples each dimension of X from each marginal independently. Because the marginal probability of each dimension of X is $P(X_i = 1) = 0.5$, the samples are in fact drawn from the set of $\{000000, 000001, \ldots, 111111\}$ uniformly random, which is far away from the distribution of training data.

4. Belief Propagation Chain

In Contrastive Divergence learning of a probabilistic graphical model, a sampling schedule is used to approximate the expectation in order to get rid of the intractable integral. This further requires the

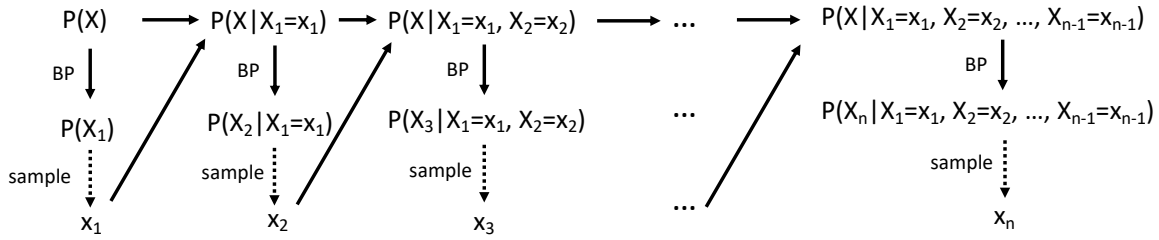


Figure 2: The procedure of running Belief Propagation Chain to obtain one sample in the iteration of Contrastive Divergence. From the joint distribution $p(X)$, we first run BP to get $p(X_1)$ and sample from it, denoting the sample x_1. Then, by fixing $X_1 = x_1$ in $p(X)$, and running BP again, we get a sample from distribution $p(X_2|X_1 = x_1)$ as x_2. Similarly, we fix $X_2 = x_2$ and run BP again, then sample from $p(X_3|X_1 = x_1, X_2 = x_2)$ to get a x_3. Keeping sampling from this conditional probability chain, finally we get a x_n sampled from $p(X_n|X_1 = x_1, X_2 = x_2, \ldots, X_{n-1} = x_{n-1})$. After concatenating these samples we can obtain a sample $x = [x_1, x_2, \ldots, x_n]$ from the joint distribution $p(X)$.

quality of the sampler to sample from the model distribution. To avoid suffering from the impact of multi-modes, we propose Belief Propagation Chain (BPChain), an algorithm which samples each dimension of the variable sequentially from a conditional distribution chain using Belief Propagation. We then equip it to Contrastive Divergence to obtain our final learning algorithm BPChain-CD.

4.1 Joint distribution as conditional probability chain

Sampling from a joint distribution $X \sim p(X_1, X_2, \ldots, X_n)$ can be viewed as sampling from a chain of conditional distributions

$$X \sim p(X_1)p(X_2|X_1)\cdots p(X_n|X_{n-1}, \cdots, X_2, X_1)$$

Based on this equation, we could firstly sample $X_1 \sim p(X_1)$. Assuming the value of $X_1 = x_1$, we then sample $X_2 \sim p(X_2|X_1 = x_1)$ and get a value of X_2 as x_2. Iteratively going along this chain we finally get all the n values by sampling n times. Then, combine them together and we get one sample from the joint distribution $p(X_1, X_2, \ldots, X_n)$. This sampling schedule makes the next sample of X independent with the previous one, which avoids the stagnate problem in Gibbs sampling. Furthermore, compared to Belief Propagation, since the production over all the conditional distributions is the joint distribution, it ensures us to deal with the situation where each dimension is correlated with each other.

4.2 Sampling by Belief Propagation Chain

In this part, we introduce BPChain to sample each dimension of the variable sequentially conditioned on the previously sampled dimensions, where BP is used to calculate each conditional marginal distribution. Though BP outputs only marginals of each variable, it can be leveraged to approximate a conditional distribution if we fix some variables with some values, e.g, if we fix $X_1 = x_1$ and run BP on the new model $p(X|X_1 = x_1; \Theta)$, it will give us the marginal distributions of from X_2 to X_N conditioned on $X_1 = x_1$.

Algorithm 1 demonstrates the procedure of drawing one sample from model distribution $p(X; \Theta)$. For each iteration i from 1 to N where N is the dimension of X, we run BP on the model

Algorithm 1 BPChain

Input: $p(X;\Theta)$, N
initialize $x \in \mathbb{R}^N$ at random
for $i = 1$ **to** N **do**
  Run BP on $p(X|X_{i-1} = x_{i-1}, \ldots, X_1 = x_1; \Theta)$
  $p(X_i;\Theta) \leftarrow$ marginal distribution of X_i after BP
  Sample $X_i \sim p(X_i;\Theta)$ to get a sample $X_i = s$
  $x_i \leftarrow s$
end for
return x

Algorithm 2 BPChain-CD

Input: $p(X;\Theta^0)$, $N, T, \eta, \{x_k\}_{k=1}^K$
for $t = 0$ **to** T **do**
  for $k = 1$ **to** K **do**
    $s_k \leftarrow$ BPChain($p(X;\Theta^t), N$)
  end for
  $\Theta^{t+1} = \Theta^t + \frac{\eta}{K}\sum_{k=1}^K(\frac{\partial \log f(x_k;\Theta^t)}{\partial\Theta} - \frac{\partial \log f(s_k;\Theta^t)}{\partial\Theta})$
end for
return $p(X;\Theta^T)$

$p(X|X_{i-1} = x_{i-1}, \ldots, X_1 = x_1; \Theta)$ where x_i means value given to random variable X_i. It will output marginal distribution of X_i conditioned on the previous sampled values, i.e, $p(X_i|X_{i-1} = x_{i-1}, \ldots, X_1 = x_1; \Theta)$. Then, after sampling a x_i from this conditional marginal and let $X_i = x_i$, we go to the next iteration. Finally, by sampling N times, we output x as one sample of random vector X. Figure 2 demonstrates this sampling process of BP Chain more specifically.

We still consider the motivating instance in Figure 1, where training data are drawn from distribution $P(X)$. Let the training model initialized as $P(X)$, at first we run BP on $P(X)$ to get $P(X_1)$ and sample from it. Assuming we have $x_1 = 1$, then after running BP on $P(X|X_1 = 1)$, we have the conditional marginal probability of X_2 as $P(X_2 = 1|X_1 = 1) = 0.4984/0.5 = 99.68\%$, which leads to $x_2 = 1$ in a large probability. Keeping sampling in this chain, we finally obtain one sample. Therefore, with BPChain sampling, we are more likely to get rid of the correlation problem of each dimension. It should be noticed that for non-binary discrete variables, BPChain can work similarly as the binary case, since multi-valued discrete variables can be represented using a few binary variables. For continuous variables, one possible solution is to discretize the continuous domain and then deal with them as non-binary discrete variables. This treatment will sacrifice some precision but is often tolerable in practice with a fine discretization.

5. Embed BPChain into Contrastive Divergence

We now present details for applying the BPChain method of the previous section to Contrastive Divergence Learning, denoted as BPChain-CD. Given the model distribution $p(X;\Theta) = \frac{1}{Z(\Theta)}f(X;\Theta)$, we take advantage of gradient descent to learning the model parameters by some training data $\{x_k\}_{k=1}^K$. Maximum-Likelihood learning is taken here and we maximize the likelihood using the Contrastive Divergence framework. In each gradient descent iteration of Contrastive Divergence, we first leverage BPChain to draw K samples from the current model, then update model parameter Θ using both training data and the drawn samples based on the equation

$$\Theta^{t+1} = \Theta^t + \frac{\eta}{K}\sum_{k=1}^K(\frac{\partial \log f(x_k;\Theta^t)}{\partial\Theta} - \frac{\partial \log f(s_k;\Theta^t)}{\partial\Theta})$$

Where x_k denotes training data and s_k denotes the drawn samples in this iteration. Algorithm 2 demonstrates the whole algorithm in detail. When Θ is updated in each iteration, we sample from a different model distribution $p(X;\Theta)$. Because BPChain leverages BP to sample from the joint distribution, with the model gradually fitting those training data, BPChain has a larger probability

to get samples similar to training data, thus make the gradient of Θ close to 0. In other words, BPChain-CD would converge where the model distribution is similar to the distribution of training data. In addition, because of better gradient descent direction raised by better samples, BPChain-CD would also converge with less iterative steps than the traditional CD algorithms.

An important aspect differentiating our technique from previous work to embed Belief Propagation (Domke, 2013) and Mean Field Network (Li and Zemel, 2014) is that our algorithm does not need to compute gradient back through the BP procedure because samples are only used to approximate the expectation and once they are sampled, they are fixed. The extra overhead lies in repeatedly running BP for all the dimensions in a single sample and running the overall procedure for all the samples. In practice, BP has been optimized for speed, making the overhead less significant in the overall execution time. This makes BPChain outperform those important sampling based algorithms like SampleSearch (Gogate and Dechter, 2011) which requires a non-trival proposal distribution and is too heavy-duty to be incorporated in CD framework. Since our method depends on BP, it inherits all the potential problems in BP like slowing learning on non-determined graph structure and numerical errors that propagate down along the chain of BPChain-CD. However, despite these problems in BP, BPChain-CD still shows better performance than those compared methods.

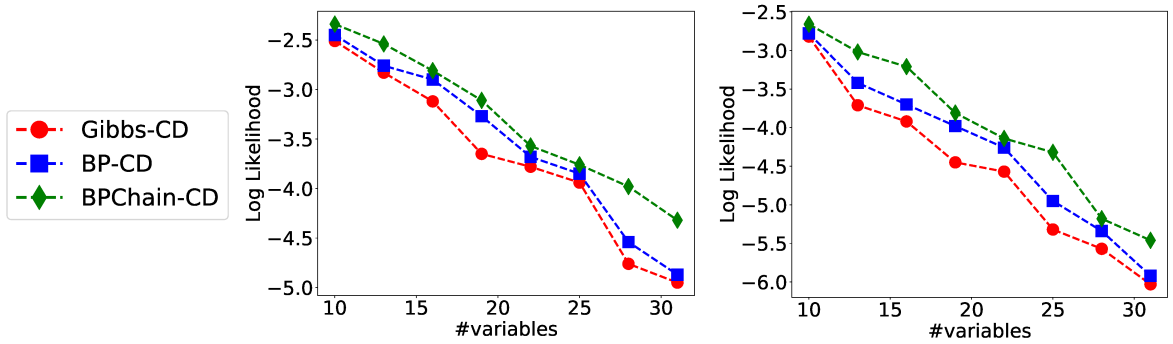
6. Experimental Results

In this section we test BPChain-CD in Algorithm 2 in three experiments. For comparison, we consider CD_{100}, denoted as Gibbs-CD, which takes advantage of Gibbs sampling (Carreira-Perpinan and Hinton, 2005) to sample the next 100 steps in each gradient descent iteration, and Belief Propagation equipped Contrastive Divergence (Ping and Ihler, 2017), denoted as BP-CD, which samples from each marginal distribution obtained by BP in each iteration. To comprehensively evaluate all algorithms, we set up a Maximum-Likelihood Learning scenario to find the probabilistic model that best fits the training dataset. To obtain the training data, we used ACE (Barton et al., 2016) to sample exactly from a target distribution. The way is to iteratively compute partition function by ACE to calculate conditional probability and sample sequentially.

We ran experiments on one node of a computing cluster with 24 cores. For each setting of all the three benchmarks, we fix the iteration step of BP in both BPChain-CD and BP-CD as 20, which is enough for BP to converge. Because samples from BPChain are more expensive, to balance time complexity during learning we draw 100 samples for BPChain-CD in each iteration while 3000 for both BP-CD and Gibbs-CD. Number of epochs is 1000. However, we run each algorithms on one cluster node with a walltime of 10 hours to make it fair comparison. We also repeat 10 times by generating 10 instances and computing the average for each setting. Our main result is shown in Figure 3,4. In a nutshell, BPChain-CD gives a better approximation to the model distribution with higher average log likelihood than that of competing approaches. In the third experiment we further show the superiority of our algorithm by precision and recall values.

6.1 Markov random field

We first consider a discrete MRF with n binary variables $x_i \in \{0, 1\}$ for $i \in \{1, \ldots, n\}$, where we represents the distribution of each clique in the discrete MRF as a table. For a MRF with n variables, we draw the number of cliques uniformly from $[n, 2n]$. Each clique c contains a subset of $\{x_i\}_{i=1}^n$ which is randomly drawn. The length of each subset is chosen from the range of $[1, 6]$ at random. In each clique c of subset length l, we have a table of size 2^l. We want to artificially create



(a) Markov random field (b) Weighted SAT

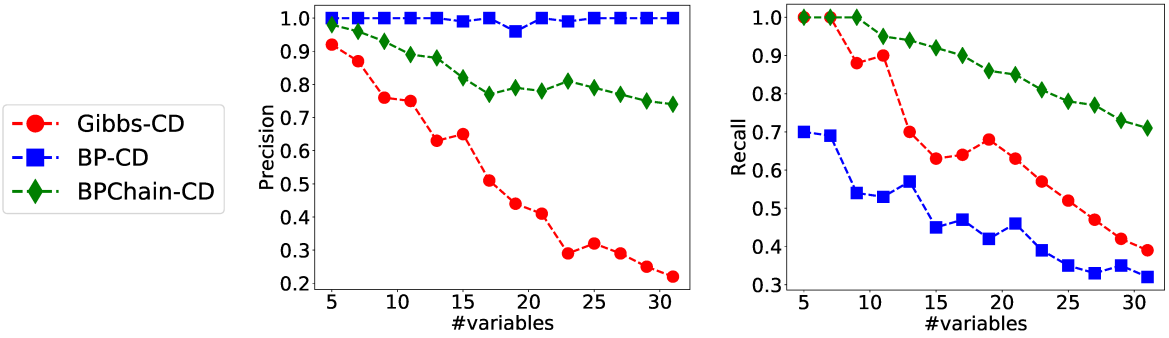
Figure 3: **(Left)** Averaged log likelihood of 100 samples with different learning algorithms on discrete MRF. **(Right)** Averaged log likelihood of 100 samples with different learning algorithms on Weighted 3-SAT. We can see that BPChain-CD learns higher average log-likelihood than competing methods BP-CD and Gibbs-CD in both of the two tasks.

a multi-modal distribution by generating many random peaks within each clique. Towards this goal, we assign a value $v = v_1 + v_2 v_3$ for each item in the clique's table. We randomly draw v_1 from $(0, 1)$, v_2 from $\{0, 1\}$, and v_3 from $[10, 1000]$. Essentially, the values v for each item correspond to a random discrete function. In the experiment, we range n from 10 to 31 in intervals of 3, and draw 10 instances for each setting. For each instance, we draw 1000 training data points (possible to overlap) for all the three algorithms. We keep the structure of model distribution the same as that we draw training data from, and initialize parameters as the absolute value of each sample from a Gaussian distribution $\mathcal{N}(10, 10)$. Learning rate is fixed as 0.1.

Figure 3(a) shows the results of the three algorithms. the x-axis is MRF with different numbers of variables, and y-axis is the average log-likelihood of randomly selected 100 samples among those training data. We can see BPChain-CD learns higher average log-likelihood of those training samples. It is because the samples generated by BPChain approximate the expectation of gradient more correctly. BP-CD is slightly better than Gibbs-CD (CD_{100}) because it generates samples in a global domain while Gibbs sampling can hardly cross the gap between multi-modes. We also find that the likelihoods of some training data from BP-CD are very high, while the others are extremely low, leading to the not high average log-likelihood in the figure. It is because BP-CD makes the model biased to some training data. We will further demonstrate it in the third experiment.

6.2 Weighted 3-SAT

We next show the performance of BPChain-CD under the setting of Weighted 3-SAT. Given n binary variables $x_i \in \{0, 1\}$ for $i \in \{1, \ldots, n\}$, we first generate random 3-SAT instances in forms of conjunctive normal form (cnf) with number of clauses $3n$. We let each clause of the cnf represented by a clique of 3 variables and a table of size 8. In each table, we replace 1 with a sample randomly drawn from $[10, 1000]$ and 0 with a sample from $[0.1, 1]$. In the experiment, we range n from 10 to 31 increased by multiples of 3 and also draw 10 instances for each setting. We also generate 1000 training data from each cnf using ACE. We keep the structure of the model distribution of the three algorithms the same as that of the cnf, where parameters are initialized from the same Gaussian distribution as in the first experiment. The other settings are also kept the same. We can see from



(a) Precision curve of generated sequences (b) Recall curve of generated sequences

Figure 4: **(Left)** Precision of all samples generated by the model learnt with different learning algorithms. **(Right)** Recall of those samples. The x-axis denotes models with different number of variables. While samples generated by the model learnt with BP-CD almost always satisfy the structured constraints, from the recall curve we can see this model is highly biased, which can only generate less than half of all the satisfied sequences with number of variables increasing. The model learnt by our algorithm is more general. It has competitive precision with that by BP-CD and is much better than Gibbs-CD, while having the highest recall of these three methods.

figure 3(b) that BPChain-CD also learns higher averaged log-likelihood than competing methods on randomly selected 100 samples among those training data. On average, the log-likelihood of the model learnt by BPChain-CD is greater than that learned by BP-CD by 0.3 and Gibbs-CD by 0.5.

6.3 Structured sequence generation

To highlight the superiority of our algorithm in fitting the training set, we also consider a generative problem under the setting of structured sequences. Consider n binary variables $x_i \in \{0, 1\}$ for $i \in \{1, \ldots, n\}$, a structured sequence is defined as follows: for $j = 0, 2, 4, \ldots, n-5$, the value set of x_j to x_{j+5} must contains three 1s and two 0s. We classify all the 2^n different sequences satisfying this structure or not as satisfied and unsatisfied. This structure makes the distribution of $x = [x_1, \ldots, x_n]$ highly joint because every two neighboring constraints share three variables. We make this rule rather than randomly select subsets because we can ensure there exist sequences satisfied under this condition. If we select subsets randomly, it is highly possible that no sequence is satisfied if we want the distribution highly joint.

In this experiment, we range the number of variables n from 5 to 31 spaced by intervals of 2 and each model having $(n-5)/2+1$ constraints. In all the 14 different models, the total number of satisfied sequences M of each model is 10, 16, 26, 41, 64, 100, 157, 247, 388, 609, 956, 1501, 2357, 3701, increasing in a rather slow speed with n. From each model we draw 3000 sequences randomly (possible to overlap) from these M satisfied sequences based on the structure defined to form the training set, and for all the three algorithms we keep our learning model the same structure of those corresponding structured sequences, i.e., for each constraint of length 5 we define a clique of those 5 variables to represent them. We initialize the model parameters as the absolute value of each sample from a Gaussian distribution $\mathcal{N}(10, 10)$. Learning rate is set as 1 at first and we

decrease it to 0.1 after 100 epochs and further decrease it to 0.01 after 500 epochs. Because of the walltime, not all algorithms could be trained to the end, so we pick the latest model to compare.

Once the training procedure finished, in each setting with the total number of satisfied sequences M we generate $5M$ samples from each learned model. Then we compute both precision and recall values. We define precision as the fraction of satisfied sequences we generate and the total number of sequences generated ($5M$), and recall as the fraction of the number of different satisfied generated sequences and the total number of satisfied sequences under this setting (M). Figure 4 shows the empirical results, where we can see from 4(a) that the model learned by BP-CD has precision 1 almost all the time, however, 4(b) tells us this model is heavily biased. By biasing on part of the solutions, the model has a large probability to generate satisfied sequences, yet could only generate as low as less than half satisfied sequences, which means that most other satisfied sequences have extremely low likelihood. Regardless of BP-CD, our algorithm is better than the traditional Gibbs-CD in both precision and recall curves.

7. Conclusion

We introduced BPChain-CD, a new variant of the Contrastive Divergence Learning framework, where samples to approximate the model distribution are generated according to a conditional distribution chain using BP. We analyzed the benefit of this sampling schedule and its significant impact on the learning process. We demonstrated that learning Discrete MRFs with this BPChain-CD could provide better results than existing CD methods on Maximum Likelihood Learning problem. It could also learn a more general model for structured sequences generation problem than Gibbs-CD and BP-CD. Future directions include a GPU-based implementation of BPChain-CD and applying the method to deep probabilistic models, such as Structure Prediction Energy Network.

8. Acknowledgement

This research was supported by the National Science Foundation (Award number IIS-1850243 and CCF-1918327). The computing infrastructure was partially supported by the Microsoft AI for Earth computing award.

References

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

J. P. Barton, E. De Leonardis, A. Coucke, and S. Cocco. Ace: adaptive cluster expansion for maximum entropy graphical model inference. *Bioinformatics*, 32(20):3089–3097, 2016.

Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural computation*, 21(6):1601–1621, 2009.

M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer, 2005.

C. Ceylan and M. U. Gutmann. Conditional noise-contrastive estimation of unnormalised models. *arXiv preprint arXiv:1806.03664*, 2018.

J. Domke. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2454–2467, 2013.

V. Gogate and R. Dechter. Samplesearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, 2011.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

J. R. Hershey, J. L. Roux, and F. Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*, 2014.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

G. E. Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

B. Jiang, T.-Y. Wu, Y. Jin, W. H. Wong, et al. Convergence of contrastive divergence algorithm in exponential family. *The Annals of Statistics*, 46(6A):3067–3098, 2018.

Y. Li and R. Zemel. Mean-field networks. *arXiv preprint arXiv:1410.5884*, 2014.

K. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. *arXiv preprint arXiv:1301.6725*, 2013.

W. Ping and A. Ihler. Belief propagation in conditional rbms for structured prediction. *arXiv preprint arXiv:1703.00986*, 2017.

M. Plummer, N. Best, K. Cowles, and K. Vines. Coda: convergence diagnosis and output analysis for mcmc. *R news*, 6(1):7–11, 2006.

F. J. Ruiz and M. K. Titsias. A contrastive divergence for combining variational inference and mcmc. *arXiv preprint arXiv:1905.04062*, 2019.

R. Salakhutdinov. Learning deep generative models. *Annual Review of Statistics and Its Application*, 2:361–385, 2015.

I. Sutskever and T. Tieleman. On the convergence properties of contrastive divergence. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 789–795, 2010.

T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

T. Tieleman and G. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1033–1040, 2009.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in neural information processing systems*, pages 689–695, 2001.