Weighted Contrastive Divergence

Enrique Romero Merino[a], Ferran Mazzanti Castrillejo[b], Jordi Delgado Pin[a], David Buchaca Prats[c]

[a]*Departament de Ciències de la Computació, Universitat Politècnica de Catalunya - BarcelonaTech, Spain*
[b]*Departament de Física i Enginyeria Nuclear, Universitat Politècnica de Catalunya - BarcelonaTech, Spain*
[c]*Barcelona Supercomputing Center (BSC), Universitat Politècnica de Catalunya - BarcelonaTech, Spain*

Abstract

Learning algorithms for energy based Boltzmann architectures that rely on gradient descent are in general computationally prohibitive, typically due to the exponential number of terms involved in computing the partition function. In this way one has to resort to approximation schemes for the evaluation of the gradient. This is the case of Restricted Boltzmann Machines (RBM) and its learning algorithm Contrastive Divergence (CD). It is well-known that CD has a number of shortcomings, and its approximation to the gradient has several drawbacks. Overcoming these defects has been the basis of much research and new algorithms have been devised, such as persistent CD. In this manuscript we propose a new algorithm that we call *Weighted CD* (WCD), built from small modifications of the negative phase in standard CD. However small these modifications may be, experimental work reported in this paper suggest that WCD provides a significant improvement over standard CD and persistent CD at a small additional computational cost.

1. Introduction

Restricted Boltzmann Machines (RBM), originally conceived in the eighties (called *harmonium* [1]) as a topological simplification of Boltzmann Machines (BM), have captured the attention of the neural network community in the last decade. This is because of its role as building blocks of multilayer learning architectures such as Deep Belief Networks (DBN) [2] or deep autoencoders [3]. RBMs have been successfully applied in several areas of interest, such as image classification [3], collaborative filtering [4] or acoustic modeling [5] to mention only a few.

An RBM is able to learn a target probability distribution from samples. RBMs have two layers, one of *hidden* and another of *visible* units, and no intra-layer connections. This property makes working with RBMs simpler than with regular BMs. This is because the stochastic computation of the log-likelihood gradient may be more efficiently evaluated, since Gibbs sampling [6] can be performed in parallel. Similar to BMs, RBM are universal approximators [7], in the sense that, given enough hidden units, RBMs can approximate any probability distribution.

In 2002, the *Contrastive Divergence* learning algorithm (CD) was put forward as an efficient training method for product-of-expert models, from which RBMs are a special case [8]. It was observed that using CD to train RBMs worked quite well in practice. This fact is important for deep learning with RBMs since some authors have suggested that a multi-layer deep neural network is better trained when each layer is separately pre-trained, as if it were a single RBM [3, 9, 10]. Thus, training RBMs with CD and stacking up RBMs is a possible way to go when designing deep learning architectures. In any case, the probabilistic potential of the RBM has been largely overlooked. More recently, RBMs have found interesting applications in solving challenging problems that are otherwise very difficult to tackle [11].

Contrastive Divergence is an approximation to the true, but computationally intractable, RBM log-likelihood gradient [12, 13]. As such, it is far from being perfect: It is biased and it may not even converge [14, 15, 16]. Also CD, and variants such as Persistent CD (PCD) [17] or Fast Persistent CD [18] can lead to a steady decrease of the log-likelihood during learning [19, 20]. Furthermore, the maximum log-likelihood models are such that the learned probability distribution accumulates most of the probability mass only on a small number of states.

In this paper we propose an alternative approximation to the CD gradient called *Weighted Contrastive Divergence* (WCD). The main difference consists in weighting the elements involved in the negative phase by its relative probability in the batch. This small but significant change leads to probability distributions that more closely resemble the target ones, at the expense of a not very large additional computational cost. In order to illustrate these points explicitly, we address small size problems that allow for an exact evaluation of both the partition function and the Kullback-Leibler divergence, which is directly related to the log-likelihood of the data. In this way we can compare the target and model probabilities of each state, thus show-

Email addresses: eromero@cs.upc.edu (Enrique Romero Merino), ferran.mazzanti@upc.edu (Ferran Mazzanti Castrillejo), jdelgado@cs.upc.edu (Jordi Delgado Pin), david.buchaca@bsc.es (David Buchaca Prats)

ing the benefits of the scheme proposed at a detailed level. We also analyze real-world large size problems, where the partition function can not be evaluated. Due to the intractability of an exact calculation of the probabilities, we use a Parzen window estimator [21] to measure the quality of the results, finding that WCD provides a significant improvement in the resulting model. A similar approach but with a different weighting scheme has recently appeared in [22].

The paper is organized as follows. Section 2 reviews the RBM model and CD. In section 3 we present the basic WCD algorithm and its natural extension, Persistent Weighted CD (WPCD). Experiments showing the benefits of WCD are described and presented in sections 4 and 5.

2. Standard CD

Binary RBMs:

$$\text{Energy}(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{b}^t\boldsymbol{x} - \boldsymbol{c}^t\boldsymbol{h} - \boldsymbol{h}^t\boldsymbol{W}\boldsymbol{x}\ , \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{h}$ are binary visible and binary hidden variables, respectively. Hidden variables are usually introduced to increase the expressive power of the model. The probability distribution of the visible variables is defined as the marginal distribution

$$P(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{h}} e^{-\text{Energy}(\boldsymbol{x},\boldsymbol{h})}}{Z}\ , \tag{2}$$

in terms of the partition function

$$Z = \sum_{\boldsymbol{x},\boldsymbol{h}} e^{-\text{Energy}(\boldsymbol{x},\boldsymbol{h})}\ . \tag{3}$$

The particular form of the energy function allows to efficiently compute the Free Energy in the numerator $e^{-\text{FreeEnergy}(\boldsymbol{x})} = \sum_{\boldsymbol{h}} e^{-\text{Energy}(\boldsymbol{x},\boldsymbol{h})}$ of Eq. (2). In addition, since both $P(\boldsymbol{h}|\boldsymbol{x})$ and $P(\boldsymbol{x}|\boldsymbol{h})$ factorize, it is possible to compute $P(\boldsymbol{h}|\boldsymbol{x})$ and $P(\boldsymbol{x}|\boldsymbol{h})$ in one step, making possible to perform Gibbs sampling efficiently [23]. However, the evaluation of Z is still computationally prohibitive when the number of input and hidden variables is large [24].

The energy function depends on several parameters $\theta = \{\boldsymbol{b},\boldsymbol{c},\boldsymbol{W}\}$. Given a data set $X = \{\boldsymbol{x}^1,\ldots,\boldsymbol{x}^N\}$, learning with RBMs consists in adjusting θ so as to maximize the log-likelihood of the data. In energy-based models, the derivative of the log-likelihood can be expressed as

$$-\frac{\partial \log P(\boldsymbol{x};\theta)}{\partial\theta} = E_{P(\boldsymbol{h}|\boldsymbol{x})}\left[\frac{\partial \text{Energy}(\boldsymbol{x},\boldsymbol{h})}{\partial\theta}\right] - E_{P(\tilde{\boldsymbol{x}})}\left[E_{P(\boldsymbol{h}|\tilde{\boldsymbol{x}})}\left[\frac{\partial \text{Energy}(\tilde{\boldsymbol{x}},\boldsymbol{h})}{\partial\theta}\right]\right] \tag{4}$$

where the first term is called the *positive phase* and the second term the *negative phase*. Similar to (2), the exact computation of the derivative of the log-likelihood is in general computationally prohibitive because the negative phase in (4) can not be efficiently computed. This is due to the fact that the negative phase comes from the derivative of the logarithm of the partition function. Notice that the log-likelihood of the data and the KLD contain the same information and can be used indistinguishably [25].

The most common learning algorithm for RBMs is called CD [8]. The algorithm for CD$_k$ estimates the derivative of the log-likelihood as

$$-\frac{\partial \log P(\boldsymbol{x};\theta)}{\partial\theta} \simeq E_{P(\boldsymbol{h}|\boldsymbol{x})}\left[\frac{\partial \text{Energy}(\boldsymbol{x},\boldsymbol{h})}{\partial\theta}\right] - E_{P(\boldsymbol{h}|\boldsymbol{x}_k)}\left[\frac{\partial \text{Energy}(\boldsymbol{x}_k,\boldsymbol{h})}{\partial\theta}\right] \tag{5}$$

where $\boldsymbol{x}_k$ is the last sample from the Gibbs chain starting from $\boldsymbol{x}$ obtained after k steps

$$\begin{aligned}
&\boldsymbol{h}_1 \sim P(\boldsymbol{h}|\boldsymbol{x})\\
&\boldsymbol{x}_1 \sim P(\boldsymbol{x}|\boldsymbol{h}_1)\\
&\ldots\\
&\boldsymbol{h}_k \sim P(\boldsymbol{h}|\boldsymbol{x}_{k-1})\\
&\boldsymbol{x}_k \sim P(\boldsymbol{x}|\boldsymbol{h}_k)\ .
\end{aligned}$$

For binary RBMs, $E_{P(\boldsymbol{h}|\boldsymbol{x})}\left[\frac{\partial \text{Energy}(\boldsymbol{x},\boldsymbol{h})}{\partial\theta}\right]$ can be easily computed and yields

- $E_{P(\boldsymbol{h}|\boldsymbol{x})}\left[\frac{\partial \text{Energy}(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{b}_j}\right] = -\boldsymbol{x}_j$
- $E_{P(\boldsymbol{h}|\boldsymbol{x})}\left[\frac{\partial \text{Energy}(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{c}_i}\right] = -\text{lgst}\left(\boldsymbol{c}_i + \boldsymbol{W}_i\ \boldsymbol{x}\right)$
- $E_{P(\boldsymbol{h}|\boldsymbol{x})}\left[\frac{\partial \text{Energy}(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{W}_{ij}}\right] = -\boldsymbol{x}_j \cdot \text{lgst}\left(\boldsymbol{c}_i + \boldsymbol{W}_i\ \boldsymbol{x}\right)$

where $\boldsymbol{W}_i$ is the i-th row of $\boldsymbol{W}$ and $\text{lgst}(z)$ stands for the logistic function $\text{lgst}(z) = \frac{1}{1+\,e^{-z}}$.

With these definitions, the modification of the weights with standard CD$_k$ for a training set $X = \{\boldsymbol{x}^1,\ldots,\boldsymbol{x}^N\}$ reads

$$\Delta(\theta) = E_{P(\boldsymbol{h}|\boldsymbol{x})}\left[\frac{\partial \text{Energy}(\boldsymbol{x},\boldsymbol{h})}{\partial\theta}\right] - \frac{1}{N}\sum_{i=1}^{N} E_{P(\boldsymbol{h}|\boldsymbol{x}_k)}\left[\frac{\partial \text{Energy}(\boldsymbol{x}_k,\boldsymbol{h})}{\partial\theta}\right]\ . \tag{6}$$

Notice that the factor $1/N$ weights equally every example in the training set, while the different probability each state should get comes from the repetition of the examples. This is important when the probabilities to be learned are non-uniform.

Nowadays the standard CD algorithm is hardly used in favor of its improved version PCD, where a continuous Markov chain is used to sample the negative phase [17].

3. Weighted Contrastive Divergence

In this section we describe the modification to the family of CD algorithms proposed in this work, WCD. First we point out the main limitations of CD, then we provide the description of the WCD algorithm, and finally present its extension to its Persistent version.

3.1. Non-desirable Behavior of Standard CD$_k$

As previously mentioned, the exact evaluation of the partition function is in general not possible. Therefore, it is difficult to compare the behavior of CD$_k$ in real world problems with respect to the exact gradient in Eq. (4). However, it is expected that the conclusions drawn for small problems, where all the probabilities can be exactly computed, may be extrapolated to large ones.

We performed an extensive evaluation of standard CD$_k$ in problems with a tractable number of input units (see section 4). From these experiments we can conclude that, in practice, standard CD$_k$ shows the following properties:

1. In many cases, standard CD$_k$ is not able to obtain good models for any combination of parameters, i.e, it is not able to assign large enough probabilities to the examples in the training set. This effect is more noticeable when k is small or the number of hidden units is small. In some cases, it happens even when k or the number of hidden units is large (see table 2), and it seems related to the difficulty of the problem.
2. For good combination of parameters, standard CD$_k$ is able to obtain somewhat good models, preserving the sum of probabilities of the training set. However, in many cases it tends to concentrate most of the probability mass in a few states, even in cases where all states in the training set should receive the same probability. As a consequence, the likelihood presents a non-monotonic behavior: it starts increasing, then reaches a maximum and starts to decrease. Accordingly, the KL divergence starts decreasing, reaches a minimum and then increases. This behaviour can be seen in several figures of section 4).

The idea behind the WCD algorithm is to (at least partially) overcome these limitations.

3.2. Description of WCD

Since the positive phase of standard CD$_k$ is exactly equal to the positive phase of the exact gradient, there is no need to modify it. In contrast, the negative phase in CD$_k$ suffers from extreme and drastic approximations that are responsible for the limitations described above. For that reason, we propose a modification of the negative phase of CD$_k$. This modification consists in weighting differently every contributing state in the negative phase. We call this new algorithm *Weighted CD*, which we describe in the following.

The negative phase of the exact gradient from Eq. (4) reads

$$\sum_{\widetilde{\boldsymbol{x}}} P(\widetilde{\boldsymbol{x}})\; E_{P(\boldsymbol{h}|\widetilde{\boldsymbol{x}})}\left[\frac{\partial \text{Energy}(\widetilde{\boldsymbol{x}},\boldsymbol{h})}{\partial\theta}\right], \tag{7}$$

and depends on all states in the space. If we consider, as usual in practice, optimization with SGA, the negative phase proposed by CD$_k$ is

$$\sum_{i=1}^{N_B} \frac{1}{N_B}\; E_{P(\boldsymbol{h}|\boldsymbol{x}_k^i)}\left[\frac{\partial \text{Energy}(\boldsymbol{x}_k^i,\boldsymbol{h})}{\partial\theta}\right] \tag{8}$$

where N_B is the number of examples in the batch. In this expression, $\boldsymbol{x}_k^i$ stands for the k-th step Gibbs-sampling reconstruction of the i-th element $\boldsymbol{x}^i$ of the batch. For small values of N_B and k (which is usually the case), it usually is a very rough approximation.

There are several important differences between the respective negatives phases in (7) and (8):

1. CD$_k$ computes the sum over the reconstructions of the data, whereas the exact gradient computes the sum over the whole space (which includes the reconstructions of the data). It means that CD$_k$ only explores, for every batch, a tiny fraction of the space. This is good from the efficiency point of view, but bad from the statistical side.
2. CD$_k$ weights every element in the sum by a weighting factor that comes from the k-step sampling distributions. However, for small k, large space size and modest batch size, repetitions in the reconstructions are highly unlikely and the implemented weighting factor is nearly the constant $\frac{1}{N_B}$, whereas the exact gradient weights every element by the model probability $P(\widetilde{\boldsymbol{x}})$. It means that CD$_k$ for low k approximately gives the same weight to every element, in contrast to the exact gradient that gives more importance to elements with larger probabilities.

The WCD algorithm proposes a way to overcome the second issue. One can assign larger weights to elements with larger probabilities by computing the relative probabilities of the elements in the batch

$$\overline{P}(\boldsymbol{x}_k^i) = \frac{P(\boldsymbol{x}_k^i)}{\sum_{j=1}^{N_B} P(\boldsymbol{x}_k^j)}, \tag{9}$$

the proposed negative phase in WCD:

$$\sum_{i=1}^{N_B} \overline{P}(\boldsymbol{x}_k^i)\; E_{P(\boldsymbol{h}|\boldsymbol{x}_k^i)}\left[\frac{\partial \text{Energy}(\boldsymbol{x}_k^i,\boldsymbol{h})}{\partial\theta}\right]. \tag{10}$$

Every weight $\overline{P}(\boldsymbol{x}_k^i)$ in (9) can be efficiently computed. First, because the partition function Z cancels out. Second, because Eq. (9) is equivalent to

$$\overline{P}(\boldsymbol{x}_k^i) = \frac{e^{\text{-FreeEnergy}(\boldsymbol{x}_k^i)}}{\sum_{j=1}^{N_B} e^{\text{-FreeEnergy}(\boldsymbol{x}_k^j)}} \tag{11}$$

while the Free Energy factorizes in the RBM topology, as previously mentioned. To summarize, in WCD one evaluates the negative phase as described in Eq. (10), while the positive phase is the same as in standard CD.

Intuitively, weighting the negative phase as in Eq. (10) allows to obtain better estimators of the real negative phase in Eq. (7), which also weights differently every state, assigning more weight to the states that have a larger Boltzmann probability. As we will confirm in the experiments with small problems, this modification has a positive effect on the learning process, allowing to obtain models with much lower KL values than standard CD$_k$. In addition, and differently from standard CD$_k$, the proposed approach fits better the training probability distribution as it is shown in section 4. Moreover, the KL has a decreasing behavior during learning. Therefore, weighting the negative phase helps overcoming the non-desired behavior of standard CD$_k$ pointed out in section 3.1. Even though such a detailed study is unfeasible with real-world large problems, the approach employed in section 5 seems to indicate that the use of a weighted negative phase also improves the statistical representativity of the model.

3.3. Generalization to Weighted Negative Phase

One can easily generalize the previous procedure to any variant of standard CD. To do so, one changes the reconstructions of the data in Eq. (10) by a suitable choice of a set of points $Y = \{\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^M\}$, leading to the *Weighted Negative Phase*, defined as

$$\sum_{i=1}^{M} \overline{P}(\boldsymbol{y}^i)\; E_{P(\boldsymbol{h}|\boldsymbol{y}^i)} \left[\frac{\partial \text{Energy}(\boldsymbol{y}^i, \boldsymbol{h})}{\partial \theta} \right] . \tag{12}$$

The previous definition does not impose any condition on the subset Y. This gives a lot of flexibility, but it also presents the additional problem of selecting a good set of candidates. In principle, any subset could be used. For instance, in the special but relevant case of PCD, Y can be taken as the set of persistent reconstructions of the data. We will denote its corresponding Weighted version as *Weighted Persistent CD* (WPCD). On the other hand, if Y spans the whole space, Eq. (12) is the negative phase of the exact gradient. Obviously, the computational cost is directly proportional to the number of elements in Y. Notice that, in general the computational overhead associated to the calculation of the negative phase in Eq. (12) is small compared with the one corresponding to the evaluation in standard CD$_k$, although this obviously depends on Y.

4. Experiments with small size problems

In the following we perform a series of experiments to test the proposed approach. More precisely, we compare standard CD$_k$ for $k = 1$ and $k = 10$, together with PCD, to their *Weighted* counterparts, WCD$_k$ and WPCD. In this section we restrict the analysis to small dimensional spaces where exact calculations can be performed. Our goal is to compare the different models at the lowest possible level and to avoid drawing conclusions from a coarse approximation, as usually done when dealing with larger problems. In particular we evaluate the exact partition function of each model, compute the exact probabilities of the whole space, and compare the probabilities of the data in the different models. We also evaluate the exact KL of the obtained models.

4.1. Data Sets

We have tested the proposed approach in a series of data sets described in the following. Training is performed including examples and probabilities. We will denote *target distribution* the set of probabilities assigned to all the states in each data set. Three different schemes have been used to establish the target distributions. The simplest one is the empirical distribution, that sets the same (uniform) probability to each state. The second one draws the probabilities from a Gaussian profile. The third model assigns different but uniform probabilities to separate subsets. We will call *training space* to the combination of data and target distribution associated to the data.

The first problem, denoted *Bars and Stripes*, consists in detecting vertical and horizontal lines in binary images containing either of them but not both. Two versions of this problem were tested, containing 3×3 (BS09) or 4×4 (BS16) images, respectively. The second problem, named *Labeled Shifter Ensemble* (LSE), consists in learning a number of states formed as follows: given an initial N-bit pattern, generate three new states concatenating to it the bit sequences 001, 010 or 100 and a new N-bit pattern computed as the original one shifting one bit to the left if the intermediate code is 001, copying it unchanged if the code is 010, or shifting it one bit to the right if the code is 100. The size of the states are $2N + 3$ bits. Again, two versions of this problem were evaluated, with $N = 4$ (LSE11) and $N = 6$ (LSE15). These problems have already been explored in [19].

The third data set tested is the Parity problem, which consists in learning whether the number of bits with value 1 is even or not. The Parity problem is known to be very difficult to learn with classical neural networks [26, 27]. It is easy to understand that this is a very difficult problem to learn in the context of Boltzmann Machines as in a high order model it requires a single weight connecting all units simultaneously [28]. This problem was tested with 8 and 10 input variables (P08 and P10, respectively).

The target distributions associated to the BS09, BS16, LSE11, LSE15, P08 and P10 problems are the correspond-

ing empirical distributions. That is, every element in each data set has a probability equal to one over the number of elements in the data set.

The fourth tested problem, which we call Int12, is a data set containing $N = 2^{12}$ integers. The unnormalized probability assigned to the each integer $n \in \{0, 1, 2 \ldots, 2^{12} - 1\}$ is given by the following expression

$$q(n) = p_{max} e^{-\lambda n^2} \ , \tag{13}$$

where λ is a parameter that depends on the maximum and minimum probabilities in the data set, p_{max} and p_{min}. The probability assigned to each element in the data set is computed as follows:

$$p(n) = \frac{q(n)}{\sum_{m=0}^{N} q(m)} \ , \tag{14}$$

where

$$\lambda = \frac{1}{(N-1)^2} \ln \left(\frac{p_{min}}{p_{max}} \right) \ .$$

The last two data sets assign different probabilities to the same 2^{12} integers, formed by the ordered list $[0, 3, 6, \ldots, 1, 4, 7, \ldots, 2, 5, 8, \ldots]$. The first variant, Mult3G, assigns to each position in the list the probability defined by Eq. (14). The second variant, Mult3D, assigns three different probability values to the elements in the list that belong to the $\dot{3}$, $\dot{3}+1$ and $\dot{3}+2$ sublists, respectively. These values are fixed imposing the sum of the probabilities in each group to be $0.6, 0.3$ and 0.1. We denote this scheme as *Discrete*. Table 1 summarizes the main properties of the training spaces used in the experiments.

4.2. Experimental Setting

The experiments were performed in two steps. In the first one we selected, for every data set, suitable parameters for standard CD_k ($k = 1$ and $k = 10$) and standard PCD. This selection was performed independently for every model. In the second step, we used the parameters found in the first step to test WCD_k and WPCD.

Networks were trained with standard gradient ascent for the BS09, BS16, LSE11, LSE15, P08 and P10 problems, and stochastic gradient ascent (with a batch size of 100) for the Int12, Mult3G and Mult3D data sets. Weights were initialized with a Gaussian distribution of zero mean and a variance that was suitably selected for every model. No weight decay was used. Every network was trained for 10^6 epochs in the BS09, BS16, LSE11, LSE15, P08 and P10 cases, and for 10^5 epochs in the Int12, Mult3G and Mult3D problems.

In order to find the optimal CD_k parameters, we performed a grid search by varying the following values:

- Number of hidden units: N_v, $2N_v$, $3N_v$, $4N_v$ and $5N_v$, where N_v is the number of visible units.
- Variances of the initial Gaussian weights: 1.0, 0.1, 0.01, 0.001 and 0.0001.
- Learning rates: 0.1, 0.01, 0.001, 0.0001 and 0.00001.

Momentum was set to 0.9. An optimal combination of parameters was selected for every k and every problem, as explained next. Every configuration of parameters was tested 10 times with different random seeds. Therefore, 1250 experiments were run for every k and every data set. Out of these experiments, we selected the combination of parameters that achieved the smallest KL at any step of the learning process.

A similar model selection was performed for PCD. The differences are described in the following. First, since we observed that for CD_k the smallest KL values were usually obtained with $5N_v$ hidden units, only this value was tested. Second, the learning rates values tested spanned the range from 10^{-1} to 10^{-8} in powers of 10 in two different schemes, fixed and linearly decaying. Third, since momentum is a relevant parameter of PCD, we have tested models with momentum values set to 0.9 and to 0.0. In the end, and as for CD_k, the optimal parameters were chosen as those that achieved the smallest KL at any step of the learning process.

After selecting the parameters for CD_1, CD_{10} and PCD, the final models were obtained for every data set in similar conditions. To that end, we tested CD_1 and WCD_1 with the parameters selected for CD_1. Furthermore, each experiment was performed varying the number of hidden units in $\{N_v, 2N_v, 3N_v, 4N_v, 5N_v\}$, where N_v is the number of visible units, and repeated 10 times with different random seeds. The same procedure was applied to CD_{10} and WCD_{10} with the parameters of CD_{10} and to PCD and WPCD with the parameters of PCD, respectively. Notice that, while the results for the weighted models may not be optimal, only better results can be achieved when their parameters are specifically optimized.

4.3. Results for the Whole Training Space

In the following we show results for the models and data sets described above when trained with the whole training space. Table 2 summarizes statistical averages over the 10 repetitions of each run, together with the standard deviations. In the table we report the mean KL obtained along the learning processes. As it can be seen, in general the Weighted version of each algorithm performs better than its non-weighted counterpart, in some cases the differences being significantly large. Notice that, as explained above, the learning parameters for WCD_k and WPCD have not even been optimized. Furthermore and as will be shown in the figures, the minimum KL in the CD and variant models is achieved at an early stage of the learning process, but afterwards degenerates. In contrast, the evolution of the KL in the *Weighted* versions is much more smooth, and the minimum KL is attained at the end of the training. Notice also that, overall, WCD_{10} is the best performer,

Data set	Input dimension	Data set size	Target distribution
BS09	9	14	Empirical
BS16	16	30	Empirical
LSE11	11	48	Empirical
LSE15	15	192	Empirical
P08	8	128	Empirical
P10	10	512	Empirical
Int12	12	4096	Gaussian Profile
Mult3G	12	4096	Gaussian Profile
Mult3D	12	4096	Discrete

Table 1: Description of the different data sets used

CD_1	CD_{10}	PCD	WCD_1	WCD_{10}	WPCD	Data set
0.0450 (0.0206)	0.0035 (0.0013)	0.0962 (0.0052)	0.0011 (0.0001)	0.0011 (0.0001)	0.0464 (0.0088)	BS09
0.1657 (0.0578)	0.0212 (0.0077)	0.1850 (0.0062)	0.0008 (0.0001)	0.0007 (0.0001)	0.1060 (0.0141)	BS16
0.2986 (0.0641)	0.0634 (0.0274)	0.1918 (0.0172)	0.1043 (0.0363)	0.0113 (0.0024)	0.1050 (0.0070)	LSE11
0.7767 (0.0913)	0.2194 (0.0441)	0.2379 (0.0136)	0.3558 (0.0888)	0.0233 (0.0021)	0.1822 (0.0071)	LSE15
0.6464 (0.0858)	0.1530 (0.0435)	0.2681 (0.0591)	0.6535 (0.0791)	0.0201 (0.0043)	0.1515 (0.0189)	P08
0.6933 (0.0001)	0.3062 (0.0169)	0.1810 (0.0456)	0.6937 (0.0019)	0.0681 (0.0232)	0.1638 (0.0313)	P10
0.0221 (0.0262)	0.0007 (0.0014)	0.0220 (0.0003)	0.0022 (0.0034)	0.0021 (0.0025)	0.0220 (0.0003)	Int12
0.5453 (0.0001)	0.0220 (0.0009)	0.1778 (0.0336)	0.0041 (0.0009)	0.0064 (0.0012)	0.1960 (0.0948)	Mult3G
0.4246 (0.0001)	0.3120 (0.1113)	0.1958 (0.0250)	0.0036 (0.0007)	0.0052 (0.0013)	0.2025 (0.0206)	Mult3D

Table 2: Mean KL values for the different problems described in the text. The standard deviations are in parenthesis.

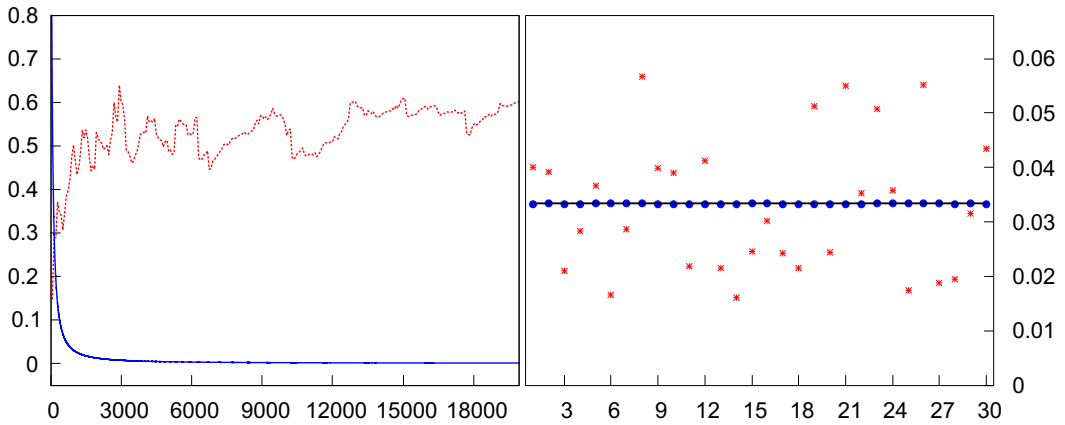

Figure 1: KL divergence during learning (left panel) and optimal probabilities (right panel) of the models found by CD_1 (red dashed line and stars) and WCD_1 (blue solid line and bullets) for the BS16 data set. The x-axis on the left panel accounts for the number of epochs/50. The x-axis on the right panel corresponds to an integer index labelling each state in the training set. The target probabilities are shown with a black line.

leading always to small KL values that are reflected in good probabilistic models.

Since we are interested in the best probability distributions, we report in the following figures the probabilities of the data in the training space for the best models out of the 10 repetitions performed in each case. These probabilities are compared with the corresponding target ones (on the right). We also report the KL values during the training process (on the left) as a way to evaluate the evolution of the models found during learning. Figure 1 shows the CD_1 and WCD_1 results obtained for the BS16 problem. As it can be seen, while the CD_1 probabilities are not dra-

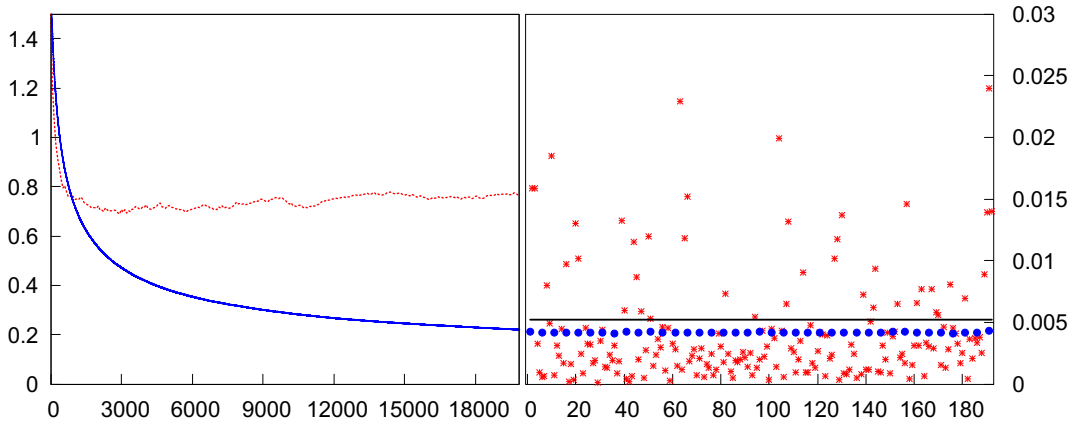

Figure 2: Same as in figure 1 for the LSE15 problem.

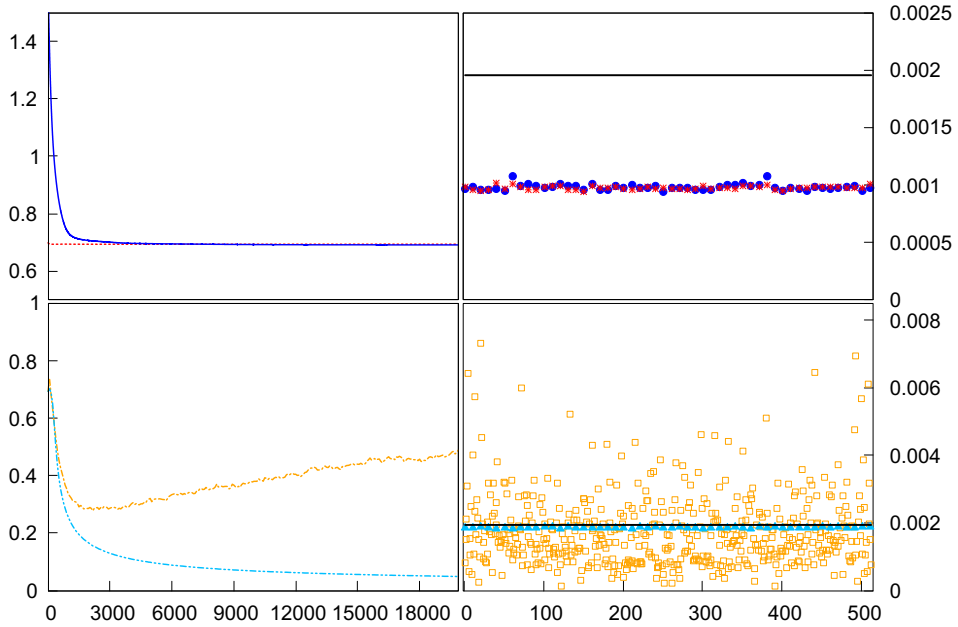

Figure 3: The top panels are the same as in figure 1 for the P10 problem. The bottom panels are the same as in figure 1 for the P10 problem in CD_{10} (orange dot-dashed line and squares) and WCD_{10} (light blue dot-dot-dashed and triangles).

matically wrong, the comparison between WCD_1 and the target probabilities is outstanding. Regarding the KL, not only the optimal and asymptotic values are much better in WCD_1, but also the models found are much more stable, according to the small local variability of the WCD_1 KL curve.

Figure 2 shows the same quantities for the LSE15 problem. In this case the evolution of the KL in CD_1 is much more smooth than in the previous case, though its optimal value found is much worse than the one found in the BS16 problem. Once again, the optimal WCD_1 KL is much lower than the corresponding CD_1 one, pointing to a better probabilistic model when compared to the target distribution. In the same way, the WCD_1 KL performs roughly as in the BS16 case, though convergence to the asymptotic value is much slower, while it still keeps its decreasing behavior. All these features are reflected in the optimal probabilities reported on the right panel. As it can be seen, the WCD_1 probabilities are much closer to the target ones than those generated by CD_1, being also much more uniform. Furthermore, the CD_1 probabilities present much larger oscillations, and in particular there are several outlayers that take a large amount of the probability mass individually, as mentioned in the introduction. In

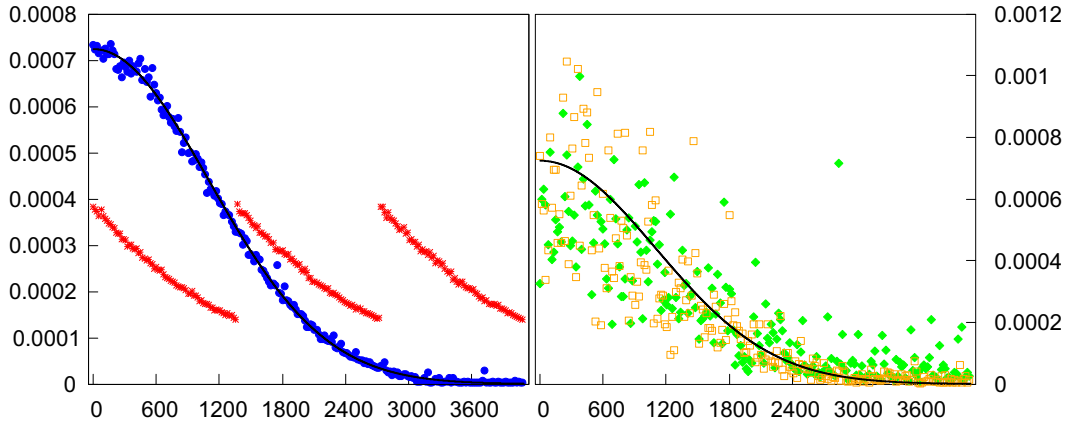

Figure 4: Optimal probabilities of the models found by CD_1 (red stars on the left panel), WCD_1 (blue bullets on the left panel), CD_{10} (orange squares on the right panel) and PCD (green diamonds on the right panel) for the Mult3G data set. The target probabilities are depicted as a black line. The x-axis on both panels is the index of every element of the data set in the corresponding ordered list (see text)

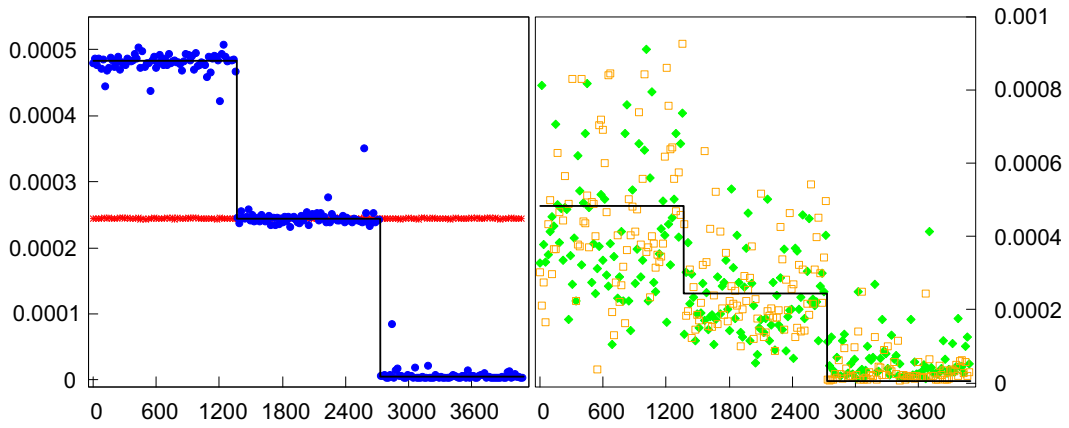

Figure 5: Same as figure 4 but for the data set Mult3D

this case, however, though the WCD_1 algorithm produces almost uniform probabilities as desired, the exact value is not reproduced as in the BS16 case, meaning that states not contained in the training space acquire non-zero probabilities. This does not happen in the WCD_{10} estimate, which achieves a very small KL value and therefore fits very well the target probabilities. As a matter of fact, this also happens in the BS16 problem, where the model probabilities already sum up to 1. The smaller version of the same problems (BS09 and LSE11) show similar behavior.

Figure 3 shows results for the toughest problem analyzed in this work, the P10 data set. The upper and lower plots show the (W)CD_1 and (W)CD_{10} results, respectively. In this case neither CD_1 nor WCD_1 are able to learn a sensible model, maybe because of the difficulty of the problem. Remarkably, the optimal CD_1 KL is almost identical to the optimal WCD_1 KL, and the evolution of the KL along learning in both cases is very similar and smooth but poor. The consequence of all this is that the resulting probability distributions are almost identical, none of them making much sense. Notice that while the probabilities found are quite uniform, they are still far away from the real target values: the training space, consisting in half the total space, receives half the total probability while it should sum up to 1. A different situation is found for the CD_{10} and WCD_{10} predictions, as shown in the lower plots in the same figure. In this case the CD_{10} KL finds a minimum but degenerates afterwards, while in the WCD_{10} case it behaves as in the previous problems, monotonically decreasing and approaching its asymptotic value, which is much lower than the CD_{10} one. The resulting probabilities are closer to the target ones in both cases, but it is remarkable how better WCD_{10} performs. While the resulting CD_1 and WCD_1 models are very similar, going from $k = 1$ to $k = 10$ leads to a much more accurate model in the WCD_{10} case, whereas CD_{10} produces once again a

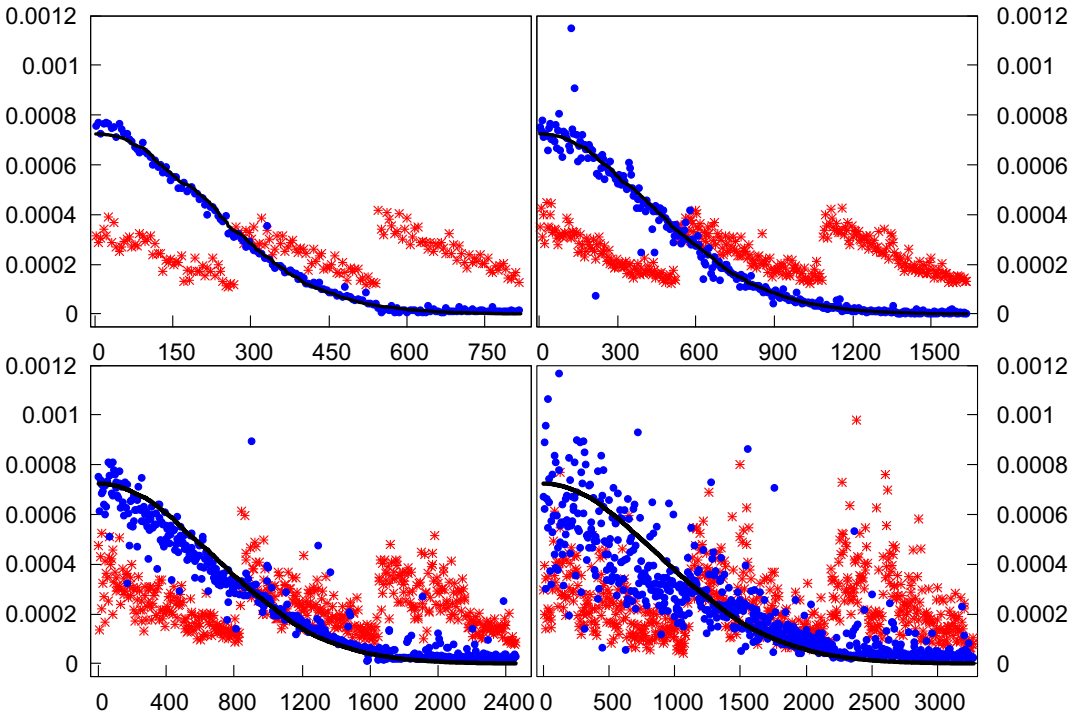

Figure 6: Probabilities of the test set corresponding to the models with minimum KL in the training set found by CD_1 (red stars) and WCD_1 (blue bullets) for the Mult3G problem. Top Left: 80%/20% training/test. Top Right: 60%/40% training/test. Bottom Left: 40%/60% training/test. Bottom Right: 20%/80% training/test. The x-axis are similar to figure 4, but only the examples in the test set are shown. The target probabilities are shown with a black line.

highly non-uniform distribution with scattered values in a broad range.

In Fig. 4 we report in two panels our results for the Mult3G problem. We have split them in two because of the different scales of the resulting probability distributions. The left panel shows CD_1 and WCD_1, while the right panel depicts the probabilities obtained in CD_{10} and PCD. As expected, the quality of the Weighted version is markedly better than the standard CD_1 prediction, the later being clearly unable to reproduce the target distribution. In the same token, WCD_1 performs better than CD_{10} and PCD. It is important to take into account that the ordering of the states in the Mult3G problem is relevant. Three different classes have been built, the first one containing the states corresponding to the values $0, 3, 6, \ldots$ in this order, the second one containing the values $1, 4, 7, \ldots$ in this order, and so on. As it can be seen, CD_1 detects that lower values have larger probabilities, but it is not able to discern among the different classes. In the CD_{10} and PCD cases, both estimations of the distribution probabilities capture the main trends of the target probabilities. However, in both cases fluctuations around the right values are large and comparable, maybe PCD performing a little bit worse, in agreement with the KL values reported in the table. On the other hand, WCD_1 once again recovers a nice model, clearly discriminating among the three groups.

Finally, in Fig 5 we report the probability distributions for the Mult3D problem obtained in CD_1, WCD_1, CD_{10} and PCD as in Fig. 4. In this case the WCD_1 prediction is dramatically better than the CD_1 one, as the later is only able to assign essentially the same probability to each state, not being able to discern any feature of the problem. In contrast, WCD_1 performs well and clearly discriminates the three categories of the problem, with quite uniform probability in each group. In We also see that CD_{10} and PCD performs similarly to the Mult3G case.

4.4. Generalization Results

The experiments described in the previous section were focused on trying to obtain the models with minimum KL in order to fit the probabilities of the training space. Therefore, they were analyzing the approximation capability of the respective models. In this section we focus on the generalization capability, which in many cases is the most important objective of the system.

To that end, we performed a number of experiments with the Int12, Mult3G and Mult3D data sets. The training set in these cases was taken to be a fraction of the whole training space used in the previous section, leaving the rest of states for the test set. The fraction values used for the training/test sets were 80%/20%, 60%/40%, 40%/60% and 20%/80%. The model with the minimum KL in the

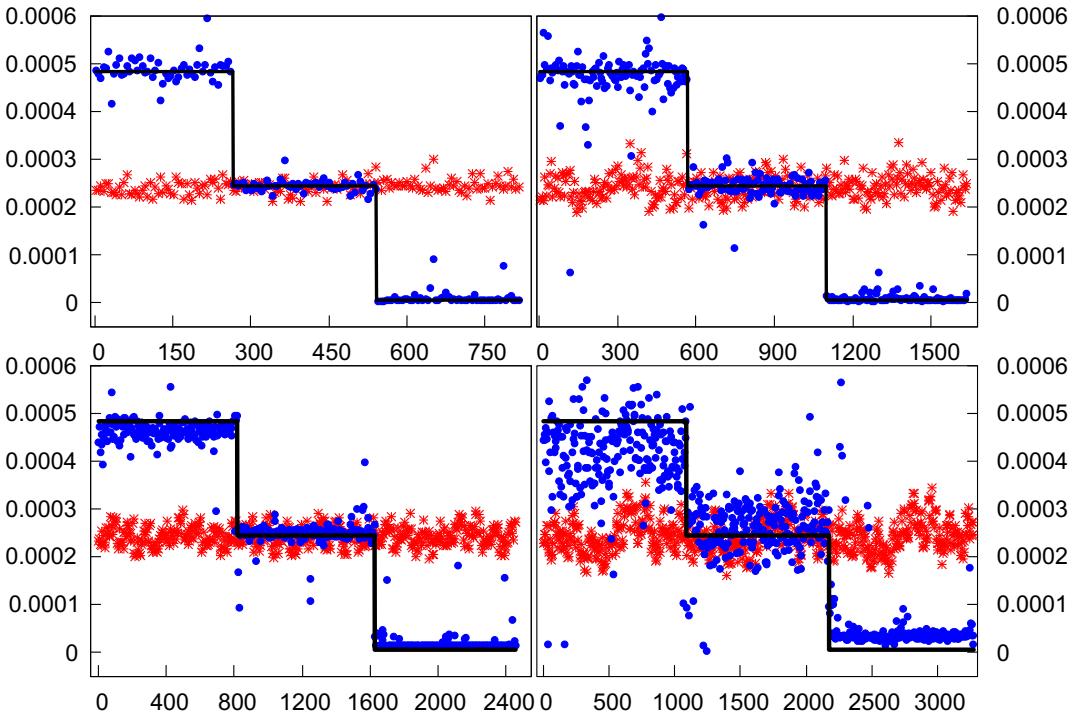

Figure 7: Same as in figure 6 for the Mult3D data set.

training set was saved and subsequently tested on the test set. As in the previous experiments, we compared standard CD_1, CD_{10}, PCD, and their *Weighted* counterparts, WCD_1, WCD_{10} and WPCD. The parameters of the models were those already selected in the experiments performed with the complete training space.

Figure 6 depicts results for the Mult3G problem. In essence, both CD_1 and WCD_1 keep the same structure observed when all states in the training space are used for learning. That means that CD_1 is always missing the main trends of the target probability, while WCD_1 keeps up fairly well. Still, the quality of WCD_1 worsens when the fraction of states used for training is reduced, as expected. But it is remarkable that, in all these cases, WCD_1 is able to generalize successfully. In particular, we notice that WCD_1 trained with just a 20% of the complete training space performs much better that bare CD_1 trained with the whole training space. Finally, figure 7 presents the same quantities as in the previous figure, for the Mult3D problem. As discussed above, this is a hard problem for CD_1 as it is never able to capture any single feature of the data. In contrast, WCD_1 holds up quite well even under severe training space restrictions. Similar results are found for the rest of the models and data sets when the training space is reduced as above.

5. Experiments with large size data sets

In this section we show how the WCD technique performs in real-world, high dimensional data sets where an exact computation of the probability of each state is unfeasible. Since the evaluation of the likelihood is not possible, we employ an alternative estimation of the quality of the results by using a Parzen window estimator [21] of the probabilities of the test set, obtained from a set of samples performed over the learned model. For computational reasons, we do not use the AIS algorithm to estimate the probabilities, which, in addition, is very hard to validate in really high dimensional spaces as the ones considered here. In short, our Parzen window estimator uses a Gaussian probability density distribution $g(\mathbf{x};\mathbf{x}_i,\sigma_i)$ centered at each point $\mathbf{x}_i$ of the sample set and with standard deviation σ_i (which we take to be the same for all points), as in [21]. The idea then is to assign a probability density at each point $\mathbf{y}_j$ in the test set equal to the averaged sum of $g(\mathbf{y}_j;\mathbf{x}_i,\sigma_i)$ over all samples. One then uses these averaged probability densities to build an unnormalized log-likelihood (uLL) of the test set, which depends on the set of samples used. More specifically, we assign to each point of the test set $\mathbf{y}_j$ an averaged Gaussian value

$$G(\mathbf{y}_j) = \frac{1}{N_s}\sum_i g(\mathbf{y}_j;\mathbf{x}_i,\sigma_i)$$

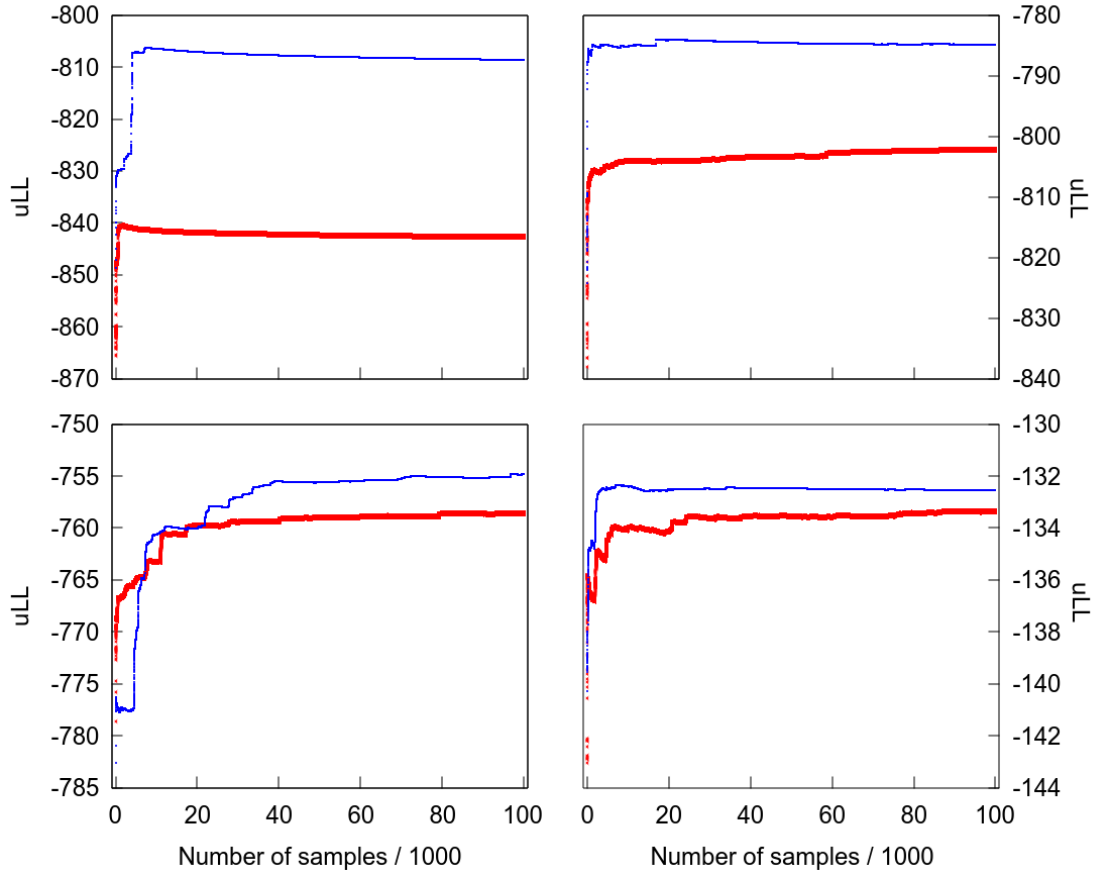

Figure 8: Comparison of the quality of the samples generated using a Parzen window procedure for WPCD (blue curves) and PCD (red curves) in for the Caltech101 (upper left), Fashion-MNIST (upper right), MNIST (lower left) and OCR-Letters (lower right) problems. The x-axis represents the number of samples generated by each model and used to compute the uLL.

where N_s is the number of samples employed, and

$$\text{uLL} = \frac{1}{N_t} \sum_j \ln G(\mathbf{y}_j) \ ,$$

with N_t the number of members of the test set. For the sake of comparison, we perform this procedure twice, using a maximum of 10^5 samples generated by the model obtained with WPCD and PCD, respectively.

We have tested the proposed approach on the following four data sets: MNIST, Fashion-MNIST, Caltech101 silhouettes and OCR letters. The MNIST data set is a well known benchmark problem corresponding to 28×28 grayscale images of hand-written digits. The fashion MNIST contains 28×28 grayscale images, associated with 10 different clothing categories (dress, coat, shirt, ...). The CalTech101 Silhouettes data set contains 28×28 binary images, containing items from 101 different categories (faces, leopards, ants, butterflies, ...). Finally, the OCR-Letters data set contains 16×8 samples of grayscale images of handwritten letters. Table 3 shows the number of features of each data set and number of samples in the train and test partitions.

In both the WPCD and PCD cases, the architecture of the network contained a variable number of visible units (depending on the problem) and a fixed number of hidden units, which was set to 500. In all cases, the networks were trained for 1000 epochs with stochastic gradient ascent and a batch size of 100 examples. Different values of the learning rate have been tested, in a logarithmic mesh spanning the range $[0.001, 1.0]$ to find the optimal value in each case. The momentum and the weight decay factors were set to 0.75 and 0.0002, respectively. Finally, the learning rate followed a linearly decaying scheme. Other combination of parameters have also been tested, to find that the best values lay in the mentioned ranges. In all cases, the selection criterion was set to get the highest test accuracy in supervised models were the labels were appended to the training examples with a one-hot coding representation. The final models were trained in an unsupervised way with the optimal values found with the model selection described above.

Figure 8 shows the comparison of the uLL of the test set as a function of the number of samples employed in the Parzen window estimator, for the four data sets an-

Table 3: Details of the data sets employed in the experiments

Dataset	Train size	Test size	Num. of features
Caltech101 Silhouette	4100	2307	784
OCR-Letters	32,152	10,000	128
MNIST	60,000	10,000	784
Fashion-MNIST	60,000	10,000	784

alyzed. In all cases, the blue and red curves correspond to the results obtained in WPCD and PCD, respectively. The upper left and right panels show the uLL for the Caltech101 and Fashion-MNIST, while the lower left and right panels correspond to the MNIST and OCR-Letters problems. As can be seen, in all cases WPCD produces higher uLL values, pointing to a better model (in a Parzen window sense) of the learned model with respect to the test set. In must be kept in mind, however, that the uLL is not a real estimation of the log-likelihood of the data, and that in all cases there is an arbitrary constant that sets the origin on the scales. That means that only the relative differences between two estimations make sense. Anyway, higher scores can be attributed to better models, although it is not possible to quantify how better. Notice that there is always a transient regime at the beginning of the curves where the variation of the uLL is large, corresponding to a poor statistical representation produced by the reduced number of samples employed. However, as this number increases, the curves approach a more stable regime where the predictions seem to stabilize, with WPCD approaching a better statistical representativity with a smaller number of samples.

6. Conclusions

In summary, in this work we propose a variant of the standard CD learning algorithm for RBMs that modifies the negative phase of the gradients involved in the weights update rule. The new negative phase is computed as a weighted average over the members of the batch, where the weighting coefficients are the relative model probabilities in the batch. This is a cheap modification that nevertheless delivers better performance, both in terms of KL and optimal model probabilities. We have tested the proposed algorithm against a set of small problems where exact probabilities can be evaluated, to find that the statistical representation of the learned models is much better than the one obtained in CD and PCD. In large problems, were a direct measure of the probabilities is unfeasible, a Parzen window evaluation of the quality of the resulting models still indicates that WPCD performs better that their alternatives. In any case, it is important to realize that, when the data can be processed and reduced to a small-dimensional space (for example, in a feature extraction stage), weighting the negative phase is a good choice that improves the probability description of the model. Furthermore, the weighting scheme can be extended to other useful techniques alternative to CD.

Possible future work involves a more sophisticated selection of elements entering in the weighted negative phase. This can, for instance, involve not only members from the training set, but also neighboring ones that, for continuity reasons, could also contain relevant information. For small problems, comparison to exact gradient calculations can also be carried out in order to contrast the statistical averages of the exact calculation to the approximated one in the learning scheme. Another interesting aspect is to analyze the convergence properties of the proposed methodology. Along this line, one can try to extend the analysis performed in [29, 30, 31], although the task is not evident for the first two since they apply to continuous units and not binary ones.

References

[1] P. Smolensky, Information Processing in Dynamical Systems: Foundations of Harmony Theory, in: D. E. Rumelhart, J. L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition (vol. 1), MIT Press, 1986, pp. 194–281.

[2] G. E. Hinton, S. Osindero, Y. Teh, A Fast Learning Algorithm for Deep Belief Nets, Neural Computation 18 (7) (2006) 1527–1554.

[3] G. E. Hinton, R. R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science 313 (5786) (2006) 504–507.

[4] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann Machines for Collaborative Filtering, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 791–798.

[5] A.-R. Mohamed, G. E. Dahl, G. Hinton, Acoustic Modeling using Deep Belief Networks, IEEE Transactions on Audio, Speech, and Language Processing 20 (1) (2012) 14–22.

[6] Y. Bengio, Learning deep architectures for AI, Foundations and Trends in Machine Learning 2 (1) (2009) 1–127.

[7] N. Le Roux, Y. Bengio, Representational Power of Restricted Boltzmann Machines and Deep Belief Networks, Neural Computation 20 (6) (2008) 1631–1649.

[8] G. E. Hinton, Training Products of Experts by Minimizing Contrastive Divergence, Neural Computation 14 (2002) 1771–1800.
[9] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy Layer-wise Training of Deep Networks, in: Advances in Neural Information Processing (NIPS'06), Vol. 19, MIT Press, 2007, pp. 153–160.
[10] H. Larochelle, Y. Bengio, J. Lourador, P. Lamblin, Exploring Strategies for Training Deep Neural Networks, Journal of Machine Learning Research 10 (2009) 1–40.
[11] G. Carleo, M. Troyer, Solving the Quantum Many-Body Problem with Artificial Neural Networks, Science 355 (6325) (2017) 602–606.
[12] Y. Bengio, O. Delalleau, Justifying and Generalizing Contrastive Divergence, Neural Computation 21 (6) (2009) 1601–1621.
[13] A. Fischer, C. Igel, Bounding the Bias of Contrastive Divergence Learning, Neural Computation 23 (3) (2011) 664–673.
[14] M. A. Carreira-Perpiñán, G. E. Hinton, On Contrastive Divergence Learning, in: International Workshop on Artificial Intelligence and Statistics, 2005, pp. 33–40.
[15] A. Yuille, The Convergence of Contrastive Divergence, in: Advances in Neural Information Processing Systems (NIPS'04), Vol. 17, MIT Press, 2005, pp. 1593–1600.
[16] D. J. C. MacKay, Failures of the one-step learning algorithm, unpublished Technical Report (2001).
[17] T. Tieleman, Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient, in: 25th International Conference on Machine Learning, 2008, pp. 1064–1071.
[18] T. Tieleman, G. E. Hinton, Using Fast Weights to Improve Persistent Contrastive Divergence, in: 26th International Conference on Machine Learning, 2009, pp. 1033–1040.
[19] A. Fischer, C. Igel, Empirical Analysis of the Divergence of Gibbs Sampling Based Learning Algorithms for Restricted Boltzmann Machines, in: International Conference on Artificial Neural Networks (ICANN), Vol. 3, 2010, pp. 208–217.
[20] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, O. Delalleau, Parallel Tempering for Training of Restricted Boltzmann Machines, in: 13th International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, pp. 145–152.
[21] O. Breuleux, Y. Bengio, P. Vincent, Quickly Generating Representative Samples from an RBM-Derived Process, Neural Computation 23 (8) (2011) 2058–2073.
[22] O. Krause, A. Fischer, C. Igel, Population-Contrastive-Divergence: Does Consistency Help with RBM training?, Pattern Recognition Letters 102 (2018) 1–7.
[23] S. Geman, D. Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (6) (1984) 721–741.
[24] P. M. Long , R. A. Serveido , RBMs are Hard to Approximately Evaluate or Simulate , in: International Conference on Machine Learning, 2010, pp. 703–710.
[25] A. C. Coolen, R. Kühn, P. Sollich, Theory of neural information processing systems, OUP Oxford, 2005.
[26] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning Internal Representations by Error Propagation, in: D. E. Rumelhart, J. L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition (vol. 1), MIT Press, 1986, pp. 318–362.
[27] Y. Bengio, Y. LeCun, Chapter 14: Scaling Learning Algorithms towards AI, in: D. D. L. Bottou, O. Chapelle, J. Weston (Eds.), Large-Scale Kernel Machine, MIT Press, 2007, pp. 321–359.
[28] E. Farguell, F. Mazzanti, E. Gomez-Ramirez, Boltzmann Machines Reduction by High-order Decimation, IEEE Transactions on Neural Networks 19 (10) (2008) 1816–1821.
[29] A. Hyvarinen, Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables, IEEE Transactions on Neural Networks 18 (5) (2007) 1529–1531.
[30] R. Karakida, M. Okada, S. ichi Amari, Dynamical analysis of contrastive divergence learning: Restricted boltzmann machines with gaussian visible units, Neural Networks 79 (2016) 78 – 87.
[31] I. Sutskever, T. Tieleman, On the convergence properties of CD , in: Y. W. Teh , M. Titterington (Eds .), Proceedings of the Thirteenth International Conference on AI and Statistics, Vol. 9 of Proceedings of Machine Learning Research, PMLR, 2010, pp. 789–795.