Restricted Boltzmann machines (RBMs) have become a popular tool for unsupervised learning in recent years. However, there still lacks an efficient algorithm for training RBMs due to the intractability of their likelihood gradient function. The existing algorithms, such as contrastive divergence and its variants, approximate the likelihood gradient using Markov chain Monte Carlo. However, the approximation error often impedes their convergence. This paper proposes a likelihood gradient free algorithm for training RBMs. The numerical results indicate that the proposed algorithm can provide a drastic improvement over the CD algorithm in RBM training. This paper also presents an extension of the proposed algorithm for how to cope with missing data in RBM training and illustrates its application using an example about drug-target interaction prediction.
A RBM is a bipartite undirected graphical model, as shown in Figure 1, which can be used to learn a probability distribution over its set of inputs. Suppose that it has $M$ visible units $v = (v_1, v_2, \ldots, v_M)$ and $N$ hidden units $h = (h_1, h_2, \ldots, h_N)$, and consists of a $N \times M$-matrix of weights $W = (w_{ij})$ associated with the connections between the hidden and visible units, as well as the bias weights $b = (b_1, b_2, \ldots, b_M)$ for the visible units and $c = (c_1, c_2, \ldots, c_N)$ for the hidden units. For the time being, we assume that the RBM is binary-binary, i.e., both the visible and hidden units take binary values. Extension of the proposed algorithm to other types of RBMs will be discussed later. For the binary-binary RBM, the joint distribution of $(v, h)$ is given by the Gibbs distribution

$$P_{\theta}(v, h) = \frac{1}{Z(\theta)} e^{-E_{\theta}(v, h)},$$

where $\theta = \{W, b, c\}$ denotes the set of parameters, $Z(\theta)$ is the partition function defined as the sum of $e^{-E_{\theta}(v, h)}$ over all possible configurations of $(v, h)$, and $E_{\theta}(v, h)$ is the energy function

$$E_{\theta}(v, h) = -\sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} v_i h_j - \sum_{j=1}^{M} b_j v_j - \sum_{i=1}^{N} c_i h_i.$$

Since there are no intra-layer connections in the RBM, the $v_j$’s are mutually independent conditioned on $h$ and conversely, the $h_i$’s are mutually independent conditioned on $v$. That is,

$$P_{\theta}(v|h) = \prod_{j=1}^{M} f_{\theta}(v_j|h), \quad \text{and} \quad P_{\theta}(h|v) = \prod_{i=1}^{N} f_{\theta}(h_i|v),$$

In this paper, we propose a gradient-free algorithm for training RBMs based on a generalization of the imputation-consistency (IC) algorithm (Liang et al., 2016). The IC algorithm was originally proposed for dealing with high-dimensional missing data problems, which works by iterating between an imputation step and a consistency step. At the imputation step, the missing data are imputed conditional on the observed data and the current estimate of parameters; and at the consistency step, a pseudo-consistent estimate of parameters is obtained by maximizing a penalized log-likelihood function of the pseudo-complete data. Under quite general conditions, it is shown that the average of the pseudo-consistent estimates is consistent to the true parameter when the number of iterations of the algorithm is sufficiently large and the data sample size is sufficiently large. However, the IC algorithm cannot be directly applied to train RBMs, as for which the log-likelihood function of the pseudo-complete data is still intractable.

To address this issue, we propose a generalized imputation-consistency (GIC) algorithm for which the consistency step is to maximize an appropriately chosen function of pseudo-complete data. Such a function should be able to drive the estimates to move toward the true parameter point with iterations, but it is not necessarily a penalized log-likelihood function of pseudo-complete data. For RBMs, we chose the function such that the parameters can be estimated by solving a sequence of logistic regressions. In this way, the new algorithm successfully avoids to approximate the likelihood gradient of the RBM. To further accelerate computation, we propose to use the coordinate descent algorithm (Tseng and Yun, 2009; Shalev-Shwartz and Tewari, 2011) to solve the logistic regressions. By employing an appropriate penalty term for the logistic regressions, the proposed algorithm provides a simple way to “drop out” redundant connections for the RBM. The numerical results indicate that the proposed algorithm can make a drastic improvement over the CD algorithm in RBM training. We also present an extension of the proposed algorithm for how to accommodate missing visible data in RBM training, and apply the extended algorithm to drug-target interaction predictions. The numerical results indicate a great success of the extended algorithm over the traditional single value decomposition (SVD) method.

2 A Likelihood Gradient Free Algorithm for RBM training

In this section, we first give a brief review for RBMs and then describe the GIC algorithm.

2.1 Restricted Boltzmann Machines

A RBM is a bipartite undirected graphical model, as shown in Figure 1, which can be used to learn a probability distribution over its set of inputs. Suppose that it has $M$ visible units $v = (v_1, v_2, \ldots, v_M)$ and $N$ hidden units $h = (h_1, h_2, \ldots, h_N)$, and consists of a $N \times M$-matrix of weights $W = (w_{ij})$ associated with the connections between the hidden and visible units, as well as the bias weights $b = (b_1, b_2, \ldots, b_M)$ for the visible units and $c = (c_1, c_2, \ldots, c_N)$ for the hidden units. For the time being, we assume that the RBM is binary-binary, i.e., both the visible and hidden units take binary values. Extension of the proposed algorithm to other types of RBMs will be discussed later. For the binary-binary RBM, the joint distribution of $(v, h)$ is given by the Gibbs distribution

$$P_{\theta}(v, h) = \frac{1}{Z(\theta)} e^{-E_{\theta}(v, h)},$$

where $\theta = \{W, b, c\}$ denotes the set of parameters, $Z(\theta)$ is the partition function defined as the sum of $e^{-E_{\theta}(v, h)}$ over all possible configurations of $(v, h)$, and $E_{\theta}(v, h)$ is the energy function

$$E_{\theta}(v, h) = -\sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} v_i h_j - \sum_{j=1}^{M} b_j v_j - \sum_{i=1}^{N} c_i h_i.$$
where $\theta_j = \{b_j, w_{ij} : i = 1, \ldots, N\}$ and $\hat{\theta}_i = \{c_i, w_{ij} : j = 1, \ldots, M\}$ denote the subsets of parameters of respective conditional distributions, and

$$f_{\theta_j}(v_j = 1|h) = \sigma(b_j + \sum_{i=1}^{N} w_{ij} h_i), \quad \text{and} \quad f_{\theta_i}(h_i = 1|v) = \sigma(c_i + \sum_{j=1}^{M} w_{ij} v_j),$$  

where $\sigma$ denotes the logistic sigmoid.

**Figure 1:** An illustrative graph of RBM with $M$ visible units (bottom row) and $N$ hidden units (upper row).

### 2.2 The GIC Algorithm

Missing data are ubiquitous throughout almost all fields of science and technology. An inappropriate treatment of missing data can lead to a significant loss of data information and/or a biased statistical inference. For low-dimensional problems, missing data can be dealt with using the EM algorithm (Dempster et al., 1977) or its variants, such as Monte Carlo EM (Wei and Tanner, 1990), ECM (Meng and Rubin, 1993), and stochastic EM (Celeux and Diebolt, 1985, Nielsen, 2000). However, for high-dimensional problems, where the data dimension is greater than the sample size, the EM algorithm and its variants often fail to work. Although some problem-specific algorithms have been developed, see e.g., Städler and Bühlmann (2012), Städler et al. (2014) and Cai et al. (2010), there still lacks a general algorithm. The GIC algorithm fills the gap: In principle, it can be applied to any missing data problems, regardless of dimension and distribution of the data. The GIC algorithm can be described as follows.

**GIC Algorithm**

Let $X_1, \ldots, X_n$ denote a random sample drawn from the distribution $f_{\theta}(x)$, where $\theta$ is a vector of parameters. Let $X_i = (X_{i}^{\text{obs}}, X_{i}^{\text{mis}})$, where $X_{i}^{\text{obs}}$ is observed and $X_{i}^{\text{mis}}$ is missed. Let $X = (X_1, \ldots, X_n), X^{\text{obs}} = (X_1^{\text{obs}}, \ldots, X_n^{\text{obs}})$ and $X^{\text{mis}} = (X_1^{\text{mis}}, \ldots, X_n^{\text{mis}})$. To indicate the dependence of the dimension of $\theta$ on the sample size $n$, we write $\theta$ as $\theta_n$ and denote by $\hat{\theta}^{(t)}_n$ the estimate of $\theta$ obtained at the $t$th iteration of the GIC algorithm. The GIC algorithm starts with an initial guess $\theta^{(0)}_n$ and then iterates between the imputation and consistency steps:

- **I-step:** Draw $\tilde{X}^{\text{mis}}$ from the predictive distribution $h(\tilde{x}^{\text{mis}}|X^{\text{obs}}, \theta^{(t)}_n)$ conditioned on $X^{\text{obs}}$ and $\theta^{(t)}_n$.
- **C-step:** Based on the pseudo-complete data $\tilde{X} = (X^{\text{obs}}, \tilde{X}^{\text{mis}})$, find an updated estimate $\theta^{(t+1)}_n$ which forms a consistent estimate of

$$\hat{\theta}^{(t)}_n = \arg \max_{\theta} E_{\theta^{(t)}_n} g(\tilde{x}, \theta),$$

where $g(\tilde{x}, \theta)$ denotes an appropriately chosen function, $E_{\theta^{(t)}_n} g(\tilde{x}, \theta) = \int g(\tilde{x}, \theta) f(x^{\text{obs}}|\theta^{*}) h(\tilde{x}^{\text{mis}}|x^{\text{obs}}, \theta^{(t)}_n) dx^{\text{obs}} d\tilde{x}^{\text{mis}}, \theta^{*}$ denotes the true value of the parameters, and $f(x^{\text{obs}}|\theta^{*})$ denote the marginal density function of $x^{\text{obs}}$. 

3
In general, the function $g(\hat{x}, \theta)$ should be chosen such that $E_{\theta^*}g(\hat{x}, \theta)$ has a unique global maximum which can be attained at the true parameter $\theta^*$, i.e., $\theta^* = \arg\max_{\theta} E_{\theta^*}g(\hat{x}, \theta)$. For example, in the original IC algorithm (Liang et al., 2016), it is set $g(\hat{x}, \theta) = \log f_{\theta}(\hat{x})$. With this choice, $\theta_n^{(t+1)}$ can be obtained by maximizing a penalized log-likelihood function of the pseudo-complete data, but which is known to be very difficult for RBMs. In this paper, we suggest to set $g(\hat{x}, \theta) = \log f_{\theta}(x^{\text{obs}}|\hat{x}^{\text{mis}})$ for RBMs, which much simplifies the computation of $\theta_n^{(t+1)}$ at each iteration. This will be detailed in Section 2.3. Under the high-dimensional scenario, i.e., the dimension of $\theta$ is much greater than $n$, we suggest to compute $\theta_n^{(t+1)}$ using a regularization approach by setting

$$\theta_n^{(t+1)} = \arg\max_{\theta} \left[ \sum_{i=1}^{n} g(\theta, x_i^{\text{obs}}, \hat{x}_i) - \lambda P(\theta) \right],$$

(5)

where $P(\theta)$ denotes a penalty function, $\lambda$ is the regularization parameter, and $\hat{x}_i^{\text{mis}}$ denotes a sample drawn from $h(x_i^{\text{mis}}|x_i^{\text{obs}}, \theta_n^{(t)})$. Based on the theory of empirical process (van de Vaart and Wellner, 1996), we proved that $\theta_n^{(t+1)}$ obtained through the regularization approach is a consistent estimate of $\theta_n^{(t)}$ under some regularity conditions, such as $n$ is sufficiently large, the parameter space $\Theta$ is compact, the function $g(\cdot)$ is well behaved, the dimension of $\theta$ grows at a rate of $O(n^{1/2})$ for some $\alpha < 1/2$, and the penalty term $\lambda P(\theta)/n$ converges to zero uniformly for all $\theta \in \Theta$. Similar to the stochastic EM algorithm, we proved that $\{\theta_n^{(t)}\}$ forms a Markov chain which converges to a stationary distribution. Further, by assuming that the mapping $M(\theta) = \arg\max_{\theta^*} E_{\theta^*}g(\theta^*, \hat{x})$ satisfies the contraction condition and the sample size $n$ is sufficiently large, we proved that $\theta_n^{(t+1)}$ will converge to the true parameter $\theta^*$ in probability when $t$ becomes large, and the mean of the stationary distribution of the Markov chain $\{\theta_n^{(t)}\}$ also converges to the true parameters $\theta^*$ in probability. Refer to the supplementary material for the detail of the proofs.

**GICC Algorithm** The GIC algorithm is attractive only when the consistent estimate of $\theta_n^{(t)}$ can be easily obtained at each C-step. We found that for many problems, similar to the ECM algorithm (Meng and Rubin, 1993), $\theta_n^{(t)}$ can be easily obtained with a number of conditional consistency steps. That is, we can partition $\theta$ into a number of blocks and then find a consistent estimate for each block conditional on the current estimates of other blocks. Suppose that $\theta = (\theta^{(1)}, \ldots, \theta^{(k)})$ has been partitioned into $k$ blocks. As an extension of the GIC algorithm, the generalized imputation-conditional consistency (GICC) algorithm can be described as follows:

- **I-step.** Draw $\hat{X}^{\text{min}}$ from the predictive distribution $h(x^{\text{min}}|X^{\text{obs}}, \theta_n^{(t,1)}, \ldots, \theta_n^{(t,k)})$ conditioned on $X^{\text{obs}}$ and the current estimate $\theta_n^{(t)} = (\theta_n^{(t,1)}, \ldots, \theta_n^{(t,k)})$.

- **CC-step.** Based on the pseudo-complete data $\hat{X} = (X^{\text{obs}}, \hat{X}^{\text{min}})$, do the following:
  1. Conditioned on $(\theta_n^{(t,2)}, \ldots, \theta_n^{(t,k)})$, find $\theta_n^{(t+1,1)}$ which forms a consistent estimate of $\theta_n^{(t,1)}$ as

$$\theta_n^{(t,1)} = \arg\max_{\theta^{(t,1)}} E_{\theta_n^{(t,1)}, \ldots, \theta_n^{(t,k)}} g_1(\hat{x}, \theta_n^{(t,1)}, \theta_n^{(t,2)}, \ldots, \theta_n^{(t,k)}),$$

for some function $g_1(\cdot)$, where the expectation $E(\cdot)$ is taken with respect to the joint distribution of $\hat{x} = (x^{\text{obs}}, x^{\text{mis}})$ and its subscript indicates the current estimate of $\theta$.

  2. Conditioned on $(\theta_n^{(t+1,1)}, \theta_n^{(t,2)}, \ldots, \theta_n^{(t,k)})$, find $\theta_n^{(t+1,2)}$ which forms a consistent estimate of

$$\theta_n^{(t,2)} = \arg\max_{\theta_n^{(t,2)}, \ldots, \theta_n^{(t,k)}} E_{\theta_n^{(t+1,1)}, \theta_n^{(t,2)}, \ldots, \theta_n^{(t,k)}} g_2(\hat{x}, \theta_n^{(t+1,1)}, \theta_n^{(t,2)}, \theta_n^{(t,3)}, \ldots, \theta_n^{(t,k)}),$$

  

$k$ Conditioned on $(\theta_n^{(t+1,1)}, \ldots, \theta_n^{(t,k-1)})$, find $\theta_n^{(t+1,k)}$ which forms a consistent estimate of

$$\theta_n^{(t,k)} = \arg\max_{\theta_n^{(t,k)}, \ldots, \theta_n^{(t+1,k-1)}} E_{\theta_n^{(t+1,1)}, \ldots, \theta_n^{(t+1,k-1)}} g_k(\hat{x}, \theta_n^{(t+1,1)}, \ldots, \theta_n^{(t+1,k-1)}, \theta_n^{(t,k)}).$$

It is easy to see that the estimate sequence $\theta_n^{(t)} = \{\theta_n^{(t,1)}, \ldots, \theta_n^{(t,k)}\}$ forms a Markov chain. Under similar conditions, we proved that the GICC algorithm shares the same theoretical properties
with the GIC algorithm; i.e., both $\theta^{(t)}_n$ and the path average form a consistent estimate of $\theta^*$ when both $f$ and $n$ are sufficiently large. Note that the functions $g_1(\cdot), \ldots, g_k(\cdot)$ used in the GICC algorithm can be different for different blocks, as long as for each of them the global maximum of the expectation function is unique and can be attained at the true parameter when the data is complete.

2.3 The GICC Algorithm for RBM Training

The task of RBM training is to estimate the parameters $\theta$ of (1). For this purpose, we treat the hidden variables $h$ as missing data, and then apply the GICC algorithm to estimate $\theta$. Suppose that $\theta$ has been partitioned into $M + 1$ blocks $(\theta_1, \theta_2, \ldots, \theta_M, c)$, where $\theta_i$’s and $c$ are as defined in Section 2.1. Let $h^{(t+1)}$ denote the imputed values of $h$ given $\theta^{(t)}$ and $v$. Set $g(\hat{x}, \theta) = \log f_\theta(x^{\text{obs}} | x^{\text{mis}})$. Then it follows from the conditional independence structure embedded in (2) that the blocks $\theta_1^{(t+1)}, \theta_2^{(t+1)}, \ldots, \theta_M^{(t+1)}$ can be calculated independently. Further, by (3), each $\theta_j^{(t+1)}$ can be calculated by solving a logistic regression where $v_j$ works as the response variable and $h^{(t+1)}$ works as the predictors. To enforce the sparsity of the RBM connections, we suggest a regularization approach which is to set

$$\theta_j^{(t+1)} = \arg \max_{\theta_j} \left[ \sum_{k=1}^{n} \log f_{\theta_j}(v_{jk}|h_k^{(t+1)}) - \lambda P(\theta_j) \right],$$

where $v_{jk}$ denotes the $k$th observation of the visible unit $j$, and $h_k^{(t+1)}$ denotes the subset of the elements of $h^{(t+1)}$ corresponding to the $k$th observation. This provides a simple way to drop out redundant connections and is expected to improve the generalization ability of the RBM. Refer to Srivastava et al. (2014) for more discussions on “dropout” methods. For (6), a variety of penalty functions can be used, such as those used in Lasso (Tibshirani, 1996), elastic net (Zou and Hastie, 2005), SCAD (Fan and Li, 2001) and MCP (Zhang, 2010), which all can lead to a consistent estimator of $\theta_j^{(t+1)}$. In this paper, we used the $L_1$-penalty in all simulations and set the regularization parameter $\lambda = 10^{-5}$. Practically, the value of $\lambda$ can be determined using a cross-validation approach, and it can be different for different blocks and at different iterations. For a given value of $\lambda$, (6) can be solved using the coordinate descent algorithm (Tseng and Yun, 2009).

Conditioned on the updated estimates $W^{(t+1)}$ and $b^{(t+1)}$, i.e., the collection of $\theta_j^{(t+1)}$’s, the parameters in $c$ can be estimated by setting

$$c_i^{(t+1)} = \arg \max_{c_i} \sum_{k=1}^{n} \log f_{\theta_i}(h_{ik}|v_k), \quad i = 1, \ldots, N,$$

The optimization problem is one-dimensional and can be easily solved by a root-finding algorithm, say, Brent’s algorithm (Brent, 1973; Press et al., 1992, p.359-362), based on its gradient. In summary, we have the following algorithm for RBM training:

- **I-step.** Draw $h^{(t+1)}$ from the distribution $P_\theta(h|v)$ as defined in (2) and (3), conditioned on $v$ and the current estimate $\theta^{(t)}$.

- **CC-step.** Based on the pseudo-complete data $(h^{(t+1)}, v)$, do the following:

  (i) For $j = 1, 2, \ldots, M$ (in parallel), calculate $\theta_j^{(t+1)}$ according to (6) by solving a penalized logistic regression using the coordinate descent algorithm.

  (ii) For $i = 1, 2, \ldots, N$ (in parallel), calculate $c_i^{(t+1)}$ according to (7) using a one-dimensional root-finding algorithm.

For the coordinate descent algorithm, we suggest to pass on the current estimate $\theta_j^{(t)}$ to the next iteration as the initial guess in calculating $\theta_j^{(t+1)}$, and this can substantially accelerate the convergence of the algorithm. In addition, $\theta_j^{(t+1)}$’s can be calculated in parallel, and $c_i^{(t+1)}$’s can also be calculated in parallel. The consistency of the algorithm follows the standard theory of the GICC algorithm. By assuming that the sample size $n$ is sufficiently large and restricting the parameter space $\Theta$ to a large compact set, $\theta_n^{(t)}$ will converge to $\theta^*$ in probability when the iteration number $t$ becomes large.
3 Numerical Studies

Example 1 We use this example, which was taken from the R package deepnet (Rong, 2015), to test the validity of the GICC algorithm for RBM training. For this example, the RBM consists of $M = 4$ visible units and $N = 2$ hidden units. The input data consists of 200 observations, with 50 observations for each of the four patterns (1,0,1,0), (0,1,1,0), (1,0,0,1), and (0,1,0,1). We are interested in this example as for which the hidden units are known to take values (1,1), (0,1), (1,0) and (0,0) (up to permutations) to represent the four patterns of the input data when the RBM is well trained.

The GICC algorithm was applied to this example, with each component of $\theta$ initialized by a random variable drawn from the Gaussian distribution $N(0, 0.1^2)$. To measure the convergence of the algorithm, we calculated the reconstruction error, which is defined as the squared difference between a data point and the “reconstruction”, i.e., the expected value of the visible nodes given the expected value of the hidden nodes and the data point. Figure 2(a) shows the convergence paths of the reconstruction error in 10 independent runs of GICC. Each run consisted of 20 iterations and cost about 3.4 seconds CPU time on a T7610 workstation of 3.6GHz. All computations reported in this paper were done on the same computer. If the code was executed in parallel on the workstation (under the OpenMP platform with 45 thresholds), the real time cost by each run was only about 0.2 seconds. We have checked the values of the hidden units obtained in each run, they all converge to one permutation of the four patterns (1,1), (0,1), (1,0) and (0,0). This indicates the validity of the GICC algorithm for RBM training. As shown by Figure 2(a), the algorithm can converge very fast. In some runs, it can converge within less than 10 iterations.

Example 2 This example is to train a RBM for a wheat line dataset, which consists of 599 wheat lines and each line is genotyped with 1279 DArT markers (Diversity Array Technology). The DArT markers take binary values, denoted by their presence or absence. In the dataset, the overall mean frequency of the allele coded as ‘1’ is 0.561, with a minimum of 0.008 and a maximum of 0.987. The dataset originally came from the International Maize and Wheat improvement center, and it can be downloaded from R package BLR (de los Campos and Rodriguez, 2015).

This is a very difficult example for RBMs, as the data does not contain obvious patterns. We fitted the data by a RBM with 300 hidden units, which consists of 385,279 ($=1279 \times 300 + 1279 + 300$) parameters. Training a RBM with such a large number of parameters is a challenging task for any gradient-based methods, including the contrastive divergence (CD) algorithm. However, the GICC algorithm works extremely well for this example. Figure 2(b) shows the convergence path of the algorithm. Only after 5 iterations, the reconstruction error has been reduced to nearly zero. The entire run consisted of 20 iterations and cost about 12 minutes in real time on the T7610 workstation with 45 thresholds running in parallel. The total CPU time was about 467 minutes. Multiple runs have been tried and the convergence paths are very similar to that shown in Figure 2(b).
For comparison, we applied the CD-10 algorithm to this example, where 10 is the number of Gibbs iterations performed at each iteration of the CD algorithm for evaluating the likelihood gradient. The CD algorithm has been implemented in the R package deepnet (Rong, 2015). In our run, we have adjusted the number of iterations such that the algorithm also cost about 467 CPU minutes at the end of the run on the same T7610 workstation. The convergence path of the algorithm is shown in Figure 2(b). The comparison indicates that GICC has made a drastic improvement over the CD algorithm in RBM training. For this example, CD-10 might fail to converge or it will take extremely long time to reach the level of reconstruction error achieved by GICC in just a few iterations.

4 Prediction of Drug-Target Interactions

Drug development is known to be expensive and time consuming. Moreover, the success rate is extremely low. Motivated by the polypharmacology property that individual drugs can interact with multiple targets, drug developers often actively seek new uses for existing drugs. This is the so-called drug repositioning strategy. Toward prediction of drug-target interactions, numerous work have been published in recent years, see e.g., Bleakly and Yamanishi (2009), Cheng et al. (2012), Wang and Zeng (2013), and Ezzat et al. (2016). Among these work, Wang and Zeng (2013) is of particular interest to us, where they formulated the problem as a collaborative filtering problem and applied the RBM to make the prediction. Collaborative filtering has become an important application of RBMs since the publication of the seminal work Salakhutdinov et al. (2007). To deal with the high proportion of missing values that are often encountered in the rating data, e.g., user’s ratings of movies, Salakhutdinov et al. (2007) proposed a population RBM model. In the model, each user has a specific RBM. Each RBM has the same number of hidden units, but it can have different numbers of visible units depending on how many ratings the user has made. Each RBM has a single training case, but their weights and biases are tied together; if two users have rated the same movies, their two RBMs must use the same weights between the hidden units and the visible unit for that movie. Salakhutdinov et al. (2007) showed that the population RBM model can be used to handle very large datasets, and it slightly outperforms carefully-tuned SVD models.

Wang and Zeng (2013) adopted the population RBM model to predict drug-target interactions, where each target corresponds to a user and each drug corresponds to a “movie”. In this paper we provide a different view to the problem: we view the drug-target interactions to be predicted as missing data. Therefore, only a single RBM is used, each target works as an independent observation, and the missing drug-target interactions can be imputed iteratively in training the RBM. Compared to the population RBM model, this model is much simpler. Let $v^{\text{mis}}$ and $v^{\text{obs}}$ denote the missing and observed parts of the visible data, respectively. Let $v^{\text{mis}}_{(t)}$ denote the imputed value of $v^{\text{mis}}$ at iteration $t$. In summary, we have the following algorithm to train RBMs with missing data:

- **I-step.** Impute missing data and hidden units:
  
  (i) Draw $v^{\text{mis}}_{(t+1)}$ conditioned on $h^{(t)}$ and the current parameter estimate $\theta^{(t)}$.
  
  (ii) Draw $h^{(t+1)}$ from the distribution $P_{\theta}(h|v)$ as defined in (2) and (3), conditioned on $\theta^{(t)}$ and $v^{\text{mis}}_{(t+1)}$.

- **CC-step.** Based on the pseudo-complete data $(h^{(t+1)}, v^{\text{obs}}, v^{\text{mis}}_{(t+1)})$, do the following:
  
  (i) For $j = 1, 2, \ldots, M$ (in parallel), calculate $\theta^{(t+1)}_j$ according to (6) by solving a penalized logistic regression using the coordinate descent algorithm.
  
  (ii) For $i = 1, 2, \ldots, N$ (in parallel), calculate $c^{(t+1)}_i$ according to (7) using a one-dimensional root-finding algorithm.

We tested the proposed algorithm on MATADOR (Günther et al., 2008), which is a manually curated online database of drug-target interactions. The dataset contains 790 drugs, 2860 protein targets, and 14,964 interactions including both direct and indirect ones. We arranged the dataset into a $2860 \times 790$ binary matrix, which is very sparse with the proportion of 1’s, indicating presence of interactions, is only 0.66%. With such sparse signals, accurate prediction of the drug-target interactions is extremely difficult. To test the proposed algorithm, we first randomly selected 360 rows (protein targets) and then randomly deleted 100 elements (drugs) from each of the selected 360 rows as missing data. Through this process, we generated 10 different training datasets. Our goal is to predict the missing
36,000 elements for each of the datasets. We tried a RBM with 75 hidden units for this problem. For each dataset, the proposed algorithm was run for 70 iterations, where the first 20 iterations were discarded for the burn-in process. The expected values of the missing visible units in the remaining 50 iterations were averaged as the predicted values. Each run cost about 35.5 minutes of real time on the T7610 workstation with 45 thresholds running in parallel. The total CPU time is about 897 minutes. To measure the performance of the proposed method, we calculated the area under the precision-recall curve. Precision is defined by TP/(TP+FP) and recall is defined by TP/(TP+FN), where TP, FP and FN are defined in Table 1 for outcomes of binary decision. Since the drug-target interactions in the dataset are rare, i.e., two classes are imbalanced, the precision-recall curve can therefore provide a better measure than the ROC curve (Davis and Goadrich, 2006) for the performance of different prediction algorithms. The results are summarized in Table 2.

Table 1: Outcomes of binary decision.

<table>
<thead>
<tr>
<th></th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive</td>
<td>TP (True Positive)</td>
<td>FP (False Positive)</td>
</tr>
<tr>
<td>Predicted Negative</td>
<td>FN (False Negative)</td>
<td>TN (True Negative)</td>
</tr>
</tbody>
</table>

Table 2: Comparison of RBM-GICC (with 75 hidden units) and SVD for prediction of drug-target interactions: the results are averaged over 10 datasets. For SVD, the default number of singular values is $k = 10$ and it allows existence of missing data. “AUC” refers to the averaged area under the precision-recall curve, and “SD” refers to the standard deviation of the average.

<table>
<thead>
<tr>
<th></th>
<th>RBM-GICC</th>
<th>SVD(k=2)</th>
<th>SVD(k=5)</th>
<th>SVD*(K=10)</th>
<th>SVD(K=50)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.787</td>
<td>0.148</td>
<td>0.171</td>
<td>0.095</td>
<td>0.073</td>
</tr>
<tr>
<td>SD</td>
<td>(0.006)</td>
<td>(0.015)</td>
<td>(0.017)</td>
<td>(0.006)</td>
<td>(0.004)</td>
</tr>
</tbody>
</table>

For comparison, we applied the singular value decomposition (SVD) method to this problem. SVD has been popularly used in collaborative filtering, which decomposes a matrix (with missing values) into two components $U$ and $V$. The singular values have been folded into these matrices. A low-rank approximation for the original matrix can then be obtained based on the decomposition with a specified number of singular values. The SVD method has been implemented in the R package recommenderlab (Hahsler and Vereet, 2016). For this example, we have tried different numbers of singular values with the results summarized in Table 2. The comparison indicates that the RBM-GICC method performs much better than the SVD method. The outperformance of the RBM-GICC method may be due to the non-linearity of the RBM model. Extension of the RBM-GICC method to general collaborative filtering problems is of great interest.

5 Discussion

We have proposed a likelihood gradient free algorithm for training RBMs based on a GICC algorithm and established the consistency of the GICC algorithm for general high-dimensional missing data problems under mild conditions. The numerical results indicate that the proposed algorithm can provide a drastic improvement over the CD algorithm in RBM training. In addition, the proposed algorithm has an automatic mechanism to drop out redundant connections for the RBM. We also gave an extension of the proposed algorithm for how to cope with missing visible data in RBM training, and illustrated its application using an example on drug-target interaction predictions.

In this paper, we considered only the RBMs with the visible units restricted to binary variables. Extension to the cases that the visible units are multinomial or Gaussian is straightforward. For the former, each parameter block $\theta_j$ can be estimated by solving a multiclass logistic regression; and for the latter, each parameter block $\theta_j$ can be estimated by solving a linear regression.

Acknowledgments

Liang’s research was supported in part by the grants DMS-1612924 and R01-GM117597. The authors thank J. Xue and B. Jia for their helpful discussions and/or graphical work.
References


