On Handling Missing Values in Data Stream Mining Algorithms Based on the Restricted Boltzmann Machine

Maciej Jaworski[1(✉)], Piotr Duda[1], Danuta Rutkowska[2], and Leszek Rutkowski[1]

[1] Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa, Poland
{maciej.jaworski,piotr.duda,leszek.rutkowski}@iisi.pcz.pl
[2] Information Technology Institute, University of Social Sciences, Łódź, Poland
drutkowska@san.edu.pl

Abstract. This paper addresses the issue of data stream mining using the Restricted Boltzmann Machine (RBM). Recently, it was demonstrated that the RBM can be useful as a concept drift detector in data streams with time-changing probability density. In this paper, we consider another problem which often occurs in real-life data streams, i.e. incomplete data. We propose two modifications of the RBM learning algorithms to make them able to handle missing values. The first one inserts an additional procedure before the positive phase of the Contrastive Divergence. This procedure aims at inferring the missing values in the visible layer by performing a fixed number of Gibbs steps. The second modification introduces dimension-dependent sizes of minibatches in the stochastic gradient descent method. The proposed methods are verified experimentally, demonstrating their usability for concept drift detection in data streams with incomplete data.

Keywords: Restricted Boltzmann Machine · Data stream mining · Missing values

1 Introduction

In recent years data stream mining became a very interesting and challenging branch of data mining [3,14–16]. In this paper, we define the data stream as a sequence of data elements

$$S = (\mathbf{s}_1, \mathbf{s}_2, \dots), \tag{1}$$

which potentially can be of infinite size. Each data element is a D-dimensional vector of binary values

$$\mathbf{s}_n = [s_{n,1}, \dots, s_{n,D}] \in \{0;1\}^D \tag{2}$$

A proper data stream mining algorithm should ensure the best trade-off between the accuracy and resources consumption. In the literature, many algorithms

T. Gedeon et al. (Eds.): ICONIP 2019, CCIS 1143, pp. 347–354, 2019.
https://doi.org/10.1007/978-3-030-36802-9_37

based on traditional machine learning or data mining tools have been proposed, e.g. neural networks with the stochastic gradient descent method [4], decision trees [10] or ensemble methods [12].

The problem of data stream mining becomes more challenging if the underlying data distribution can change over time [17]. In this paper, we focus on the issue of applying the Restricted Boltzmann Machine (RBM) to detect possible changes in the data distribution. This idea was first proposed in [8], and extended in [9] to allow dealing with labeled data. In [11] the resource-awareness of the RBM in data stream scenario was investigated. In this paper, we continue the topic by proposing modifications of the RBM learning algorithm to handle data streams with missing values.

The RBM is a special type of a wider class of neural networks called Boltzmann Machines [7]. It consists of two layers of neurons: the visible one, consisting of D neurons $\mathbf{v} = [v_1, \ldots, v_D]$ and the hidden one, which is formed by H hidden units $\mathbf{h} = [h_1, \ldots, h_H]$. For each possible state $(\mathbf{v}, \mathbf{h})$ of the RBM an energy can be calculated, which is defined as follows

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{D} v_i a_i - \sum_{j=1}^{H} h_j b_j - \sum_{i=1}^{D}\sum_{j=1}^{H} v_i h_j w_{ij}, \tag{3}$$

where w_{ij}, a_i and B_j are RBM weights and biases. The energy function is used to define a probability distribution of $(\mathbf{v}, \mathbf{h})$

$$P(\mathbf{v}, \mathbf{h}) = \frac{\exp\left(-E(\mathbf{v}, \mathbf{h})\right)}{Z}, \tag{4}$$

where Z is a normalization constant. Let us assume that the data stream (1) is partitioned into minibatches of size B, i.e. the t-th minibatch is given by

$$S_t = \left(\mathbf{s}_{Bt+1}, \ldots, \mathbf{s}_{Bt+B}\right), \ t = 0, 1, \ldots. \tag{5}$$

Then, the cost function for S_t is given by the following formula

$$C(S_t) = -\log P(S_t) = -\frac{1}{B}\sum_{n=1}^{B}\sum_{\mathbf{h}} \log P(\mathbf{v} = \mathbf{s}_{Bt+n}, \mathbf{h}) \tag{6}$$

and its gradient with respect to weight w_{ij} is expressed as follows (see. e.g. [2,4])

$$\frac{\partial C(S_t)}{\partial w_{ij}} = \sum_{\mathbf{v},\mathbf{h}} P(\mathbf{v}, \mathbf{h}) v_i h_j - \frac{1}{B}\sum_{n=1}^{B}\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v} = \mathbf{s}_{Bt+n}) v_i h_j. \tag{7}$$

The first term on the right-hand side ('negative phase'), is intractable to compute and can be approximated by the Contrastive Divergence (CD) algorithm [5]. In this paper we propose some modifications of the CD algorithm, allowing the RBM to handle incomplete data.

The rest of the paper is organized as follows. In Sect. 2 the CD algorithm for learning the RBM is recalled. It is shown how it is used for approximating the gradient of the RBM cost function. In Sect. 3 two modifications are proposed which allow the RBM to handle incomplete data. Preliminary results of experimental verification of presented methods are demonstrated in Sect. 4. Conclusions are discussed in Sect. 5.

2 Contrastive Divergence Learning Algorithm

As can be seen in (7), the gradient of the cost function $\frac{\partial C}{\partial w_{ij}}$ consists of two terms. Each term is based on sampling from different probability distributions. The second term, called the 'positive phase', requires the procedure of inferring the states of the hidden units from the data element, which is presented in Algorithm 1.

Algorithm 1: Hidden layer inference based on a data element

infer($\mathbf{s}$):
$\mathbf{v} \leftarrow \mathbf{s}$;
for $j \leftarrow 1$ **to** H **do**
  $h_j \leftarrow P(h_j|\mathbf{v})$;
end

The first term of gradient $\frac{\partial C}{\partial w_{ij}}$, called the 'negative phase', is intractable to compute. In the CD algorithm, it is approximated by performing a Gibbs sampling algorithm [1], presented in Algorithm 2.

Algorithm 2: Gibbs sampling

GibbsSampling(K):
for $k \leftarrow 1$ **to** K **do**
  for $i \leftarrow 1$ **to** D **do**
    $v_i \leftarrow P(v_i|\mathbf{h})$;
  end
  for $j \leftarrow 1$ **to** H **do**
    $h_j \leftarrow P(h_j|\mathbf{v})$;
    if $k < K$ **then**
      $h_j \leftarrow 1$ with prob. h_j, otherwise $h_j \leftarrow 0$;
    end
  end
end

Algorithm 3: Gradients updating

updateGradients(sgn):
for $i \leftarrow 1$ **to** D **do**
  for $j \leftarrow 1$ **to** H **do**
    $\frac{\partial C}{\partial w_{ij}} \leftarrow \frac{\partial C}{\partial w_{ij}} + sgn\frac{1}{B}v_i h_j$;
  end
end
for $i \leftarrow 1$ **to** D **do**
  $\frac{\partial C}{\partial a_i} \leftarrow \frac{\partial C}{\partial a_i} + sgn\frac{1}{B}v_i$;
end
for $j \leftarrow 1$ **to** H **do**
  $\frac{\partial C}{\partial b_j} \leftarrow \frac{\partial C}{\partial b_j} + sgn\frac{1}{B}h_j$;
end

Algorithm 4: The Contrastive Divergence algorithm (CD)

CD(S_t, K):
$\frac{\partial C}{\partial w_{ij}} = 0$, $\frac{\partial C}{\partial a_i} = 0$, $\frac{\partial C}{\partial b_j = 0}$, $i = 1, \ldots, D$, $j = 1, \ldots, H$;
for $\boldsymbol{s} \in S_t$ **do**
  infer(**s**);
  updateGradients(-1);
  GibbsSampling(K);
  updateGradients(1);
end

For both phases the gradient values can be updated using the procedure presented in Algorithm 3 (where $sgn = -1$ and $sgn = 1$ correspond to the positive and negative phases, respectively). Finally, the CD algorithm for minibatch S_t, consisting of all mentioned previously components, is presented in Algorithm 4.

3 RBM for Handling Incomplete Data

Algorithm 5: Missing values restoring

Restore(**s**,Q,**m**):
$\mathbf{v} \leftarrow s$;
for $q \leftarrow 1$ **to** Q **do**
  for $j \leftarrow 1$ **to** H **do**
    $h_j \leftarrow P(h_j|\mathbf{v})$;
    $h_j \leftarrow 1$ with prob.
    h_j, otherwise
    $h_j \leftarrow 0$;
  end
  for $i \leftarrow 1$ **to** D **do**
    if $m_i == TRUE$
    then $v_i \leftarrow P(v_i|\mathbf{h})$;
  end
end
Return **v**;

Algorithm 6: Gradients updating with the masks of missing values taken into account

updateGradientsMasked(sgn, **m**, **B**):
for $i \leftarrow 1$ **to** D **do**
  for $j \leftarrow 1$ **to** H **do**
    if $m_i == FALSE$ **then** $\frac{\partial C}{\partial w_{ij}} \leftarrow \frac{\partial C}{\partial w_{ij}} + sgn \frac{1}{B_i} v_i h_j$;
  end
end
for $i \leftarrow 1$ **to** D **do**
  if $m_i == FALSE$ **then** $\frac{\partial C}{\partial a_i} \leftarrow \frac{\partial C}{\partial a_i} + sgn \frac{1}{B_i} v_i$;
end
for $j \leftarrow 1$ **to** H **do**
  $\frac{\partial C}{\partial b_j} \leftarrow \frac{\partial C}{\partial b_j} + sgn \frac{1}{B} h_j$;
end

In the practical guide for training RBMs [6] several methods for inferring missing values were proposed. However, none of them seems to work fast enough to be suitable for data stream mining tasks. In the sequel, we propose two modifications of the CD algorithm to make it able to handle data streams with missing values.

For each minibatch of data elements S_t we assume that there exists a minibatch of masks $M_t = (\mathbf{m}_{Bt+1}, \ldots, \mathbf{m}_{Bt+B})$. Each mask $\mathbf{m}_n$ is a D-dimensional

vector of $\{TRUE, FALSE\}$ values. If $m_{n,i}$ is $TRUE$, then the value of $s_{n,i}$ is unknown. When necessary, by default this value is assumed to be equal to 0, until it is not restored. The first modification of the basic CD algorithm is to introduce a restoring function, presented in Algorithm 5. This procedure performs Gibbs sampling, however, only unknown units of the visible layer are updated.

The second proposed modification changes the gradients updating method. In the basic CD method, updates of gradients are calculated as the arithmetic average over the whole minibatch of data (as in Algorithm 3). In our approach, we introduce variable-sized minibatches. The size of the minibatch for the i-th dimension is equal to the number of data elements, for which the mask in the i-th dimension is $FALSE$

$$B_i(M_t) = \sum_{n=1}^{B} 1_{\{m_{Bt+n,i}==FALSE\}}. \tag{8}$$

Let $\mathbf{B} = (B_1, \ldots, B_D)$ be a D-dimensional vector of dimension-dependent minibatch sizes. Then the method for gradients update, which takes the missing values into account, is presented in Algorithm 6.

The final form of the Contrastive Divergence algorithm for data with missing values, which we abbreviate here as CDM, is presented in Algorithm 7.

Algorithm 7: The Contrastive Divergence algorithm for data with missing values (CDM)

```
CDM(S_t, M_t, K, Q, Rest, PosMask, NegMask):
∂C/∂w_ij = 0, ∂C/∂a_i = 0, ∂C/∂b_j=0, i = 1,...,D, j = 1,...,H;
B = 0;
for m ∈ M_t do
    for i ← 1 to D do
        if m_i == FALSE then B_i++;
    end
end
for (s, m) ∈ (S_t, M_t) do
    if Rest == TRUE then s = Restore(s, Q, m);
    infer(s);
    if PosMask == TRUE then updateGradientsMasked(−1, m, B);
    else updateGradients(−1);
    GibbsSampling(K);
    if NegMask == TRUE then updateGradientsMasked(1, m, B);
    else updateGradients(1);
end
```

Comparing to the standard CD algorithm it requires several additional arguments. These are the minibatch of masks M_t, corresponding to the minibatch of data S_t, and the number of steps Q of the restoring procedure. Three last parameters, i.e. $Rest$, $PosMask$, and $NegMask$ are boolean flags, which allow to turn on or off previously discussed modifications in the CDM algorithm.

4 Experimental Results

In this section, we present some preliminary results of the experimental verification of the presented methods. The numerical simulations were carried out on the MNIST dataset [13]. It contains 60000 gray-scale images of handwritten digits of size 28×28. In experiments, we treat the dataset as a stream. The data order is mixed randomly. Then, it is processed with minibatches of size $B = 20$. For each data element, a mask of missing values was assigned. The mask was in the form of a square of size $z \times z$ pixels. The position of this square on the image was chosen randomly, with equal probability for each possible location. The parameters for learning the RBM were set as follows: $D = 784$, $H = 40$, $K = 1$, $Q = 1$, the learning rate $\eta = 0.05$. We applied standard stochastic gradient method with momentum – the friction parameter was equal to $\gamma = 0.9$.

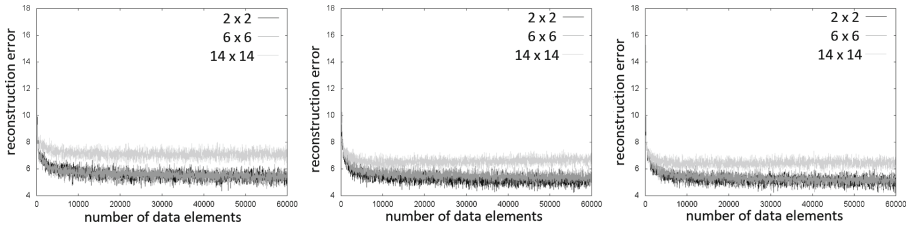
Looking at Algorithm 7, one can see that there are many possible variants of the proposed CDM algorithm. In the simulations we focus on three of them together with the standard CD algorithm:

- CD: $Rest = FALSE$, $PosMask = FALSE$, $NegMask = FALSE$;
- CDM(TFF): $Rest = TRUE$, $PosMask = FALSE$, $NegMask = FALSE$;
- CDM(TTT): $Rest = TRUE$, $PosMask = TRUE$, $NegMask = TRUE$;

Algorithms were evaluated in the prequential manner using the reconstruction error. For the considered minibatch of data S_t a set of reconstructions $\tilde{S}_t = (\tilde{\mathbf{s}}_{Bt+1}, \ldots, \tilde{\mathbf{s}}_{Bt+B})$ has to be obtained first using the RBM. Then, the average reconstruction error is expressed as follows

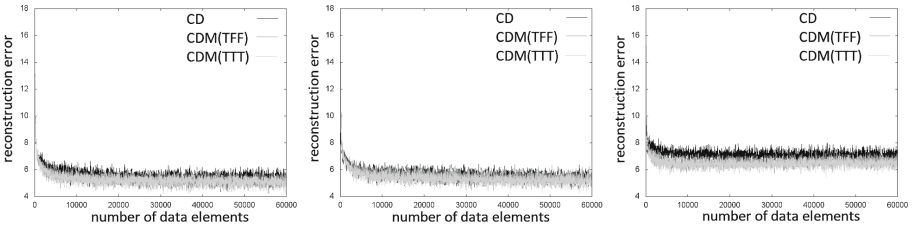
$$R(S_t) = \frac{1}{B} \sum_{n=1}^{B} \sum_{i=1}^{D} \left(s_{Bt+n,i} - \tilde{s}_{Bt+n,i} \right)^2 . \tag{9}$$

In the first experiment, the considered algorithms were run with three various sizes of missing values masks: $z = 2$, $z = 6$ and $z = 14$. The comparison of each algorithm performance for various values of z is demonstrated in Fig. 1. As can be seen, for each algorithm the reconstruction error is positively correlated with the amount of noise in data elements. Let us now look at the results of this experiment in another configuration. In Fig. 2 the algorithms are compared for each considered value of z. Although the values of reconstruction error fluctuate significantly in each case, it is possible to notice that the algorithm with all considered previously mechanisms turned on (i.e. the CDM(TTT) algorithm) is slightly better than the two others, whereas the standard CD algorithm is always the worst. It is the most clearly seen for the case with the biggest noise (i.e. $z = 14$). Although the differences are not striking, it can be concluded that the proposed modifications improve the performance of the CD algorithm when the incomplete data have to be handled.



a) CD b) CDM(TFF) c) CDM(TTT)

Fig. 1. Reconstruction error obtained for various sizes of missing values masks for three considered algorithms.



a) $z = 2$ b) $z = 6$ c) $z = 14$

Fig. 2. Reconstruction error of the CD, CDM(TFF) and CDM(TTT) algorithms for three different sizes of missing values masks.

5 Conclusions

In this paper, we considered the problem of mining stream data with missing values using the Restricted Boltzmann Machine (RBM), focusing our analysis on the Contrastive Divergence (CD) algorithm. To make it able to handle incomplete data, we proposed two modification. The first one is to introduce an additional Gibbs sampling procedure at the beginning of processing each data element. However, only those units of the visible layer are updated for which the value of the corresponding dimension in the data element is missing. In the second modification, the fixed size of minibatch is replaced by minibatches with dimension-dependent sizes. This means that not all data from the minibatch take part in updating gradients of RBM weights or visual layer biases. The proposed methods were verified experimentally, demonstrating their usability for concept drift detection in data streams with incomplete data.

Acknowledgments. The project financed under the program of the Minister of Science and Higher Education under the name "Regional Initiative of Excellence" in the years 2019–2022 project number 020/RID/2018/19, the amount of financing 12,000,000 PLN. This work was also supported by the Polish National Science Centre under grant no. 2017/27/B/ST6/02852.

References

1. Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I.: An introduction to MCMC for machine learning. Mach. Learn. **50**(1), 5–43 (2003)
2. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
3. Devi, V.S., Meena, L.: Parallel MCNN (PMCNN) with application to prototype selection on large and streaming data. J. Artif. Intell. Soft Comput. Res. **7**(3), 155–169 (2017)
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
5. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Comput. **14**(8), 1771–1800 (2002)
6. Hinton, G.E.: A practical guide to training restricted Boltzmann machines. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 599–619. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_32
7. Hinton, G.E., Sejnowski, T.J., Ackley, D.H.: Boltzmann machines: Constraint satisfaction networks that learn. Technical report CMU-CS-84-119, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA (1984)
8. Jaworski, M., Duda, P., Rutkowski, L.: On applying the restricted Boltzmann machine to active concept drift detection. In: Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence, Honolulu, USA, pp. 3512–3519 (2017)
9. Jaworski, M., Duda, P., Rutkowski, L.: Concept drift detection in streams of labelled data using the restricted Boltzmann machine. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–7 (2018)
10. Jaworski, M., Duda, P., Rutkowski, L.: New splitting criteria for decision trees in stationary data streams. IEEE Trans. Neural Netw. Learn. Syst. **29**(6), 2516–2529 (2018)
11. Jaworski, M., Rutkowski, L., Duda, P., Cader, A.: Resource-aware data stream mining using the restricted Boltzmann machine. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2019. LNCS (LNAI), vol. 11509, pp. 384–396. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20915-5_35
12. Krawczyk, B., Cano, A.: Online ensemble learning with abstaining classifiers for drifting and noisy data streams. Appl. Soft Comput. **68**, 677–692 (2018)
13. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010). http://yann.lecun.com/exdb/mnist/
14. Lemaire, V., Salperwyck, C., Bondu, A.: A survey on supervised classification on data streams. In: Zimányi, E., Kutsche, R.-D. (eds.) eBISS 2014. LNBIP, vol. 205, pp. 88–125. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17551-5_4
15. Ramirez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., Herrera, F.: A survey on data preprocessing for data stream mining: current status and future directions. Neurocomputing **239**, 39–57 (2017)
16. Rutkowski, L., Jaworski, M., Duda, P.: Stream Data Mining: Algorithms and Their Probabilistic Properties. SBD, vol. 56. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-13962-9
17. Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 27–39 (2014)