Low-cost Representation for Restricted Boltzmann Machines

Son N. Tran and Artur d'Avila Garcez

Department of Computer Science
City University London, UK
Son.Tran.1@city.ac.uk,aag@soi.city.ac.uk

Abstract. This paper presents a method for extracting a low-cost representation from restricted Boltzmann machines. The new representation can be considered as a compression of the network, requiring much less storage capacity while reasonably preserving the network's performance at feature learning. We show that the compression can be done by converting the weight matrix of real numbers into a matrix of three values $\{-1,\ 0,\ 1\}$ associated with a score vector of real numbers. This set of values is relatively similar to Boolean values which help us further translate the representation into logical rules. In the experiments reported in this paper, we evaluate the performance of our compression method on image datasets, obtaining promising results. Experiments on the MNIST handwritten digit classification dataset, for example, have shown that a 95% saving in memory can be achieved with no significant drop in accuracy.

Keywords: Restricted Boltzmann Machines, Low-cost Representation, Knowledge Extraction

1 Introduction

Restricted Boltzmann Machines (RBMs) [8, 1] are a generative model which can learn interesting hidden features from data. In many applications, RBMs have been shown advantageous over traditional feature extraction at training classifiers, especially when RBMs are stacked onto a deep network to form, e.g. a Deep Belief Network [2]. However, due to their structural complexity, these feature learning models require a large storage of memory. In this paper, we propose a method for extracting a low-cost representation from RBMs, as a step towards the use of Deep Belief Networks in memory-limited devices. The low-cost representation is expected to require less memory for storage, while reasonably preserving the performance of the RBMs. Furthermore, we show that our low-cost representation can be translated into a logic-like language, thus providing an intuitive understanding of the data and being compatible with boolean circuits.

We are concerned with the use of RBMs as feature extractors whereby the hidden features are generated from a logistic function of the weighted combination of the original features, obtained from a dataset. For the low-cost representation, we use the same logistic function with some changes to the weights.

In particular, we convert the weight matrix of real values from an RBM into a matrix where each entry has three possible states $\{-1, 0, 1\}$, and each column vector has a real-valued score associated with it. Since there are only three possible values for each element of the matrix, one needs to use only two bits to represent them. Furthermore, by removing any rows where the low-cost matrix has value zero, a further compression can be achieved. The result, as we shall see, is that the relationships among the input variables in an RBM can be represented by logic rules, similarly to [10]. Experiments on the MNIST handwritten digit classification dataset have shown that a 95% saving in memory can be achieved with no significant drop in accuracy, and up to 99% saving can be achieved with the low-cost feature extraction still offering a significant improvement on the baseline SVM classification applied directly to the input data.

The remainder of the paper is organized as follows. In Section 2, we present background on RBMs. In Section 3, we present the low-cost representation and the network-compression algorithm. In Section 4, we recall the relationship between the compressed representation and logic. Section 5 contains experimental results on the MNIST and related datasets, and Section 6 concludes the paper and discusses directions for future work.

2 Background

A Restricted Boltzmann Machine [8] is a two-layer symmetric connectionist system with no connections between units in the same layer. We use V and H to denote, respectively, the visible and hidden layers of an RBM. We use $W \in \mathbb{R}^{I \times J}$, where I is the number of visible units and J is the number of hidden units, to denote the RBM's weight matrix, with w_{ij} denoting the connection weight from visible unit i to hidden unit j. The energy function of a network with states of visible layer $V = v$ and hidden layer $H = h$ is given by:

$$\mathbf{E}(v,h) = -\sum_{ij} v_i w_{ij} h_j - \sum_i a_i v_i - \sum_j b_j h_j \tag{1}$$

Here, w_{ij}, a_i, b_j are the connection weights, biases for visible units, and biases for hidden units, respectively. The joint distribution of the network's states is:

$$P(v,h) = \frac{e^{-\mathbf{E}(v,h)}}{Z} \tag{2}$$

with $Z = \sum_{v,h} e^{-\mathbf{E}(v,h)}$. Given the state of a layer, the units in the other layer are conditionally independent and can be sampled from the following distributions:

$$\begin{aligned} P(v_i|h) &= \sigma(\sum_j w_{ij} h_j + a_i) \\ P(h_j|v) &= \sigma(\sum_i w_{ij} v_i + b_j) \end{aligned} \tag{3}$$

with $\sigma(x) = \frac{1}{1+e^{-x}}$, called a logistic function.

Training RBMs is difficult due to the computational intractability of the partition function Z. However, one can use efficient approximation methods such as Contrastive Divergence [1] to estimate such parameters reasonably well.

In previous work, [10], we have shown that the memory cost of an RBM can be reduced by pruning low-scoring feature detectors. In what follows, we show that the memory cost can be further reduced by converting the weight matrix into a three-valued matrix and a vector of scores.

3 Low-cost Representation for RBMs

RBMs have been used as a powerful tool for the extraction of features from a dataset. Normally, the RBM is used to perform a non-linear transformation of the data from its original space v to the space of hidden variables h. The probability f_j of unit h_j in the hidden layer being activated given an input v, is given by (from Eq. (3)):

$$f_j = \sigma(\mathbf{w}_j^\top v + b_j) \tag{4}$$

where $\mathbf{w}_j$ is column vector j in the weight matrix W of the RBM, and is also known as a basis vector or feature detector.

We now propose a function to transform the features from the data space to the same hidden space as follows:

$$f'_j = \sigma(c_j \mathbf{s}_j^\top v + b_j) \tag{5}$$

where c_j is a real value and $\mathbf{s}_j \in \{-1,\ 0,\ 1\}^I$ is a low-cost vector having the same size as $\mathbf{w}_j$, with element s_{ij} having one of the values -1, 0, or 1. In order to make our proposed features f'_j useful, we need to be able to extract c_j and $\mathbf{s}_j$ from the feature detector $\mathbf{w}_j$ of the RBM such that f'_j approximates f_j. We do this by minimizing the squared Euclidean distance between the basis vector $\mathbf{w}_j$ and the low-cost vector $\mathbf{s}_j$ weighted by c_j, as follow:

$$d(\mathbf{w}_j, c_j\mathbf{s}_j) = \frac{1}{I}\sum_i \|w_{ij} - c_j s_{ij}\|_1^2 \tag{6}$$

Note that (6) is quadratic, the optimal value of c_j can be found by setting the derivatives of the squared Euclidean to zeros, such that:

$$c_j = \frac{\sum_i w_{ij}s_{ij}}{\sum_i s_{ij}^2} \tag{7}$$

Since the value of s_{ij} is in the set $\{-1, 0, 1\}$, we have:

$$\begin{aligned}\|w_{ij} - c_j s_{ij}\|_1^2 &= \|\text{abs}(w_{ij}) - c_j \frac{s_{ij}}{\text{sign}(w_{ij})}\|_1^2 \\ &= \begin{cases} (\text{abs}(w_{ij}) + c_j)^2 & \text{if } s_{ij} \neq \text{sign}(w_{ij}) \\ (\text{abs}(w_{ij}) - c_j)^2 & \text{if } s_{ij} = \text{sign}(w_{ij}) \\ \text{abs}(w_{ij})^2 & \text{if } s_{ij} = 0 \end{cases}\end{aligned} \tag{8}$$

Here, $\mathrm{abs}(w_{ij})$ and $\mathrm{sign}(w_{ij})$ are functions that return the absolute value and sign of w_{ij}, respectively. Since $(\mathrm{abs}(w_{ij}) + c_j)^2 > (\mathrm{abs}(w_{ij}) - c_j)^2$ and $(\mathrm{abs}(w_{ij}) + c_j)^2 > \mathrm{abs}(w_{ij})^2$, $s_{ij} = 0$ will minimize the Euclidean distance if and only if $\mathrm{abs}(w_{ij})^2 \leq (\mathrm{abs}(w_{ij}) - c_j)^2$ from which $c_j \geq 2 \times \mathrm{abs}(w_{ij})$.
We are now able to describe the procedure to extract c_j and $\mathbf{s}_{ij}$ from the RBM, as follows:

Step 1: Initialize $\mathbf{s}_j$ so that $s_{ij} = \mathrm{sign}(w_{ij})$ [1]
Step 2: For each hidden node j, compute c_j using Eq. (7)
Step 3: For each connection weight w_{ij}, set $s_{ij} = 0$ if $c_j \geq 2 \times \mathrm{abs}(w_{ij})$
Step 4: Compute c_j; if c_j is unchanged then stop, otherwise go to Step 3.

4 Relation to Logic Representation

In this section, we show that the use of low-cost vectors is similar to the confidence logic representation from [9, 10]. A confidence-based logic formula is a logic programming (*if-then*) implication of the form $c : \mathsf{h} \leftarrow \mathsf{b}_1 \wedge \cdots \wedge \mathsf{b}_n$, where h is a logical atom and each b_i, $1 \leq i \leq n$, is a logical literal (an atom or its negation), labelled by a real-valued number c called a *confidence-value.* For example: $1.5 : \mathsf{h} \leftarrow \mathsf{b}_1 \wedge \neg\mathsf{b}_2 \wedge \mathsf{b}_3$ associates hypothesis (hidden unit) h with beliefs (visible units) b_1, *not* $\mathsf{b}_2, \mathsf{b}_3$ with confidence value 1.5.
If we remove every s_{ij} whose $c_j = 0$ from the low-cost vector $\mathbf{s}_j$ then we are able present each function $f'_j = \sigma(c_j \mathbf{s}_j^\top v + b_j)$ in the following confidence-logic form:

$$c_j : f'_j \leftarrow \bigwedge_{s_{i'j}=1} v_{i'j} \wedge \bigwedge_{s_{i''j}=-1} \neg v_{i''j} \tag{9}$$

In what follow, we present two examples, using the XOR function and the MNIST images data set, to illustrate the above translation.

Example 1. XOR function

We trained an RBM with 10 hidden units on the truth-table of the XOR function with 3 variables X, Y, Z. Suppose that we would like Z to be our target variable (notice that we could have equally chosen X or Y without retraining the model). Below, we present the confidence-logic rules in which literal h_j appears together with target literal z or $\neg\mathsf{z}$. By combining rules of the form $\mathsf{h} \leftarrow \mathsf{z}$ and $\mathsf{z} \leftarrow \mathsf{h}$ into $\mathsf{h} \leftrightarrow \mathsf{z}$, we obtain the set of rules below; treating h_j as an intermediate concept and combining each pair of rules to obtain rules relating x, y and z directly, and ignoring the confidence-values, one obtains the four rules for XOR, e.g., from $\mathsf{h}_2 \leftarrow \neg\mathsf{x} \wedge \neg\mathsf{y}$ and $\neg\mathsf{z} \leftrightarrow \mathsf{h}_2$, one obtains $\neg\mathsf{z} \leftarrow \neg\mathsf{x} \wedge \neg\mathsf{y}$.

$6.843 : \mathsf{h}_2 \leftarrow \neg\mathsf{x} \wedge \neg\mathsf{y}; 4.008 : \neg\mathsf{z} \leftrightarrow \mathsf{h}_2;$ $5.342 : \mathsf{h}_3 \leftarrow \mathsf{x} \wedge \mathsf{y}; 4.008 : \neg\mathsf{z} \leftrightarrow \mathsf{h}_3$
$3.984 : \mathsf{h}_5 \leftarrow \neg\mathsf{x} \wedge \neg\mathsf{y}; 4.008 : \neg\mathsf{z} \leftrightarrow \mathsf{h}_5;$ $2.668 : \mathsf{h}_6 \leftarrow \mathsf{x} \wedge \mathsf{y}; 4.008 : \neg\mathsf{z} \leftrightarrow \mathsf{h}_6$
$4.611 : \mathsf{h}_7 \leftarrow \neg\mathsf{x} \wedge \mathsf{y}; 4.008 : \mathsf{z} \leftrightarrow \mathsf{h}_7;$ $2.389 : \mathsf{h}_8 \leftarrow \mathsf{x} \wedge \mathsf{y}; 4.008 : \neg\mathsf{z} \leftrightarrow \mathsf{h}_8$
$3.847 : \mathsf{h}_9 \leftarrow \mathsf{x} \wedge \neg\mathsf{y}; 4.008 : \mathsf{z} \leftrightarrow \mathsf{h}_9;$ $4.015 : \mathsf{h}_{10} \leftarrow \neg\mathsf{x} \wedge \mathsf{y}; 4.008 : \mathsf{z} \leftrightarrow \mathsf{h}_{10}$

Example 2. Handwritten Characters

We have applied the same process from the previous example to the MNIST dataset. We trained a sparse RBM[4] with 500 hidden units and 794 visible units (784 pixel variables and 10 softmax class variables). Below, we present a visualization of the confidence-logic rules whereby a positive literal v_i is shown as a white pixel, a negative literal $\neg v_i$ is shown as a black pixel, and a literal that does not appear in the rule is shown as a grey pixel. Normally, the rules are organized, by the way in which they are obtained, into two levels: one with relations between pixel variables and hidden variables, and another with relations between hidden variables and target variables. For ease of presentation, we omit the scores (confidence-values) from the first level, and also omit the negative literals from the second level, before we replace the hidden literals (intermediate concepts) with the visible literals for visualization. Because of space restrictions, we only show 6 images per rule for 5 (out of 10) rules.

5 Experimental Results

We performed experiments with the MNIST handwritten digits dataset[2], TiCC handwritten characters dataset[3] and YALE face dataset[4]. In each dataset, we divide the data into training, validation and test set. For the MNIST dataset, we use a subset of the training data with $10,000$ samples (MNIST$_{10K}$), 2000 validation samples, and $10,000$ test samples for a digits recognition task (from 0 to 9). We also use the same test set to test the representation extracted from RBMs trained on the entire training set with $60,000$ samples[5] (MNIST$_{60K}$). The TiCC dataset consists of $18,189$ training samples, $1,250$ validation samples, and $18,177$ test samples for a person's letter recognition task (from A to Z). We divide the YALE dataset into a training set with 135 samples, thus 9 samples per person, and the test set with 30 samples. We used an SVM with Gaussian kernel

[1] $s_{ij} = 0$ if $w_{ij} = 0$

[2] http://yann.lecun.com/exdb/mnist/

[3] http://algoval.essex.ac.uk:8080/icdar2005/index.jsp?page=ocr.html

[4] http://vision.ucsd.edu/content/yale-face-database

[5] Here, we re-use the hyper-parameters from the experiment with $10,000$ training samples.

as a classifier to measure the performance of the extracted low-cost representation in comparison with the RBMs. Model selection is performed by running a grid-like search (except for the YALE dataset) over the learning rates for the RBMs (between 0.001 and 1), cost (between 0.0001 and 100), and gamma (between 0.0001 and 100)) for the SVM, all on a log-scale. We did not select the number of hidden units in the RBMs, instead we tested RBMs with 500 and 1000 hidden units only, simply to investigate whether the size of the network affects the quality of the extracted low-cost representation. All the results using the MNIST dataset can be reproduced using the MATLAB code provided at https://github.com/sFunzi/Low-costRBM/. The reader can contact the authors directly if interested in the results obtained using the other datasets.

The memory needed by each type of representation, i.e. RBMs and our low-cost representation, can be defined as follows:

$$\begin{aligned} M_{RBM} &= T \times C_{word} \times I \times J \\ M_{low-cost} &= (2 \times I \times J) + (T \times C_{word} \times J) \end{aligned} \tag{10}$$

where C_{word} is the number of bits of a computer word in a device, and T is the number of computer words of a real-valued data type. For example, in a 32-bit machine, a RBM with 784 visible units and 500 hidden units will cost $2 \times 32 \times 784 \times 500 = 25,088,000$ bits for a double precision floating point type. In the case of an implementation of low-cost vectors in a computing device which needs 2 bits to represent an element in the vector then the memory cost should be $(2 \times 784 \times 500) + (2 \times 32 \times 500) = 816,000$ bits. The ratio of memory saved by the low-cost representation over the RBM can be measured by:

$$r_{save} = \frac{M_{RBM} - M_{low-cost}}{M_{RBM}} \times 100\% \tag{11}$$

	float	double
r_{save} no pruning	93.622%	96.747%
r_{save} 20% pruning	94.898%	97.398%
r_{save} 40% pruning	96.173%	98.048%
r_{save} 60% pruning	97.449%	98.699%
r_{save} 80% pruning	98.724%	99.349%

Table 1: The expected memory saving ratios for an RBM with 784 visible units and 500 hidden units using standard floating point data types in a 32-bit computer; *pruning* refers to the percentage of low-scoring hidden nodes removed from the RBM

In our experiments, we have trained RBMs using double-precision floating point weight matrices on a 32-bit computer in order to evaluate the performance of the low-cost representation in comparison with that of the original RBMs at performing feature extraction. Hence, our purpose is to compare the accuracy and ratio of memory saved of the RBMs and their low-cost representation. We assume the existence of a feasible hardware implementation of the low-cost representation. We also investigate how accuracy drops as the RBMs are pruned, in comparison with pruning of their low-cost representation with respect to the ratio of memory saved. Pruning of x% of a network (RBM or its low-cost counterpart) means that the x% lowest-scoring vectors $\mathbf{s}_j$, i.e. with the smallest values of c_j, are removed, as done in [10].

Table 2 contains the accuracies of the RBMs with 500 and 1000 hidden nodes trained on four datasets, and the accuracies of their low-cost counterparts, all on the held-out test sets. We have run each experiment 10 times and report the mean accuracy, along with standard deviation. The results show that the performance of the low-cost representation can be almost identical to that of the RBMs, with high consistency.

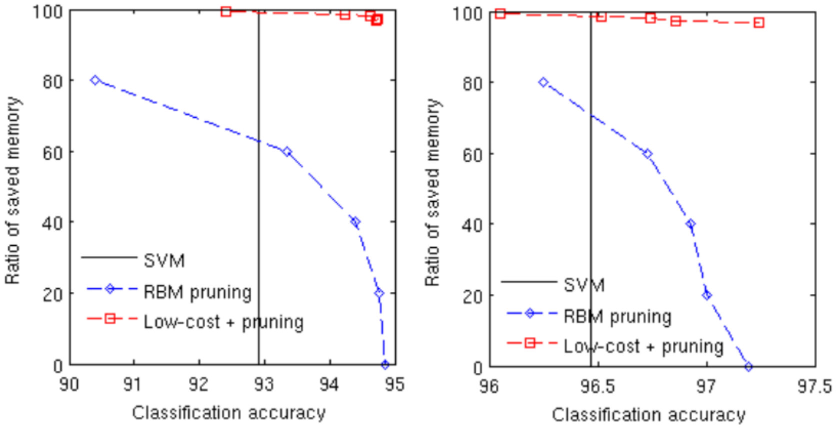
	TiCC	MNIST$_{10K}$	MNIST$_{60K}$	YALE face
RBM (J=500)	94.851% ± 0.033	97.198 ± 0.060	98.553% ± 0.031	95.000% ± 2.833
Low-cost	94.711% ± 0.072	97.240 ± 0.089	98.530% ± 0.040	94.333% ± 3.865
RBM (J=1000)	94.928% ± 0.016	97.245% ± 0.031	98.680% ± 0.024	97.000% ± 2.919
Low-cost	94.729% ± 0.070	97.219% ± 0.056	98.562% ± 0.035	96.667% ± 1.757

Table 2: Average test set performance of RBMs in comparison with their low-cost representation on four different datasets

Next, we evaluate the effectiveness of the low-cost representation in comparison with pruning the RBM. For both the RBM and its low-cost counterpart, one can rank and remove the low-scoring vectors $\mathbf{s}_j$, for which c_j is relatively low. For the sake of comparison, we prune $20\%, 40\%, 60\% and 80\%$ of both the RBMs and the low-cost representation, and evaluate performance. As expected, the average test set error increases with the pruning. However, results show that more than 98% memory saving can be achieved by the low-cost representation with the feature extraction still offering a significant improvement on the baseline SVM classification obtained from the input data directly.

In order to show the usefulness of the compressed representation at feature extraction, we use the classification accuracy obtained by an SVM on the original input data as baseline. We found that for the MNIST$_{60K}$ and YALE face datasets, the features extracted by either the RBM or the low-cost representation produced only a slight improvement on the original data trained using an SVM. In the experiments with the TICC and MNIST$_{10K}$ datasets, however, feature extraction outperformed the SVMs. Therefore, we have chosen the latter two datasets to visualize and evaluate the effect of pruning, as shown in Figure 1 for RBMs containing 500 hidden units only.

In Figure 1, the SVM line indicates the test set error on the raw input data without using RBMs at all. This line separates the space into an area where the use of an RBM, low-cost or otherwise, can improve performance (on the left hand side) and an area where feature extraction, whichever the memory capacity gains, is not warranted (on the right hand side). Notice that, in the case of the MNIST dataset, since a 0.2% increase in accuracy is generally accepted as a significant improvement [3], Figure 1 shows that approximately 98% of memory capacity



(a) TiCC dataset (b) MNIST dataset

Fig. 1: Error rate progression in comparison with memory capacity gains for RBMs and low-cost RBMs pruned by 0, 20, 40, 60 and 80%

gains can be obtained from storing a low-cost RBM for feature extraction, while preserving a significant improvement over the baseline SVM classification applied to the raw input data.

6 Conclusions and Future Work

We have presented a method for the extraction of a low-cost representation from restricted Boltzmann machines, which may be seen as a step towards the integration of deep networks in memory limited devices. The new representation offers a compression of the network, which theoretically requires less storage memory, while preserving to some extent most of the network's performance at feature learning. In the experiments reported in this paper, it is shown that the low-cost representation proposed here is advantageous over RBMs in terms of memory efficiency. The experiments also indicate that the performance of the low-cost RBMs is almost identical, in practice, or acceptably lower than that of the full RBMs. As future work, we intend to consider details of hardware implementation and a real application using lower-memory devices.

References

1. Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002.
2. Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comp.*, 18(7):1527–1554, July 2006.
3. Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, page 536543, New York, NY, USA, 2008. ACM.

4. Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area v2. In *Advances in Neural Information Processing Systems.* MIT Press, 2008.
5. Leo de Penning, Artur S. d'Avila Garcez, Lus C. Lamb, and John-Jules Ch Meyer. A neural-symbolic cognitive agent for online learning and reasoning. In *IJCAI*, pages 1653–1658, 2011.
6. Gadi Pinkas. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77(2):203 – 247, 1995.
7. Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107136, February 2006.
8. Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Volume 1: Foundations*, pages 194–281. MIT Press, Cambridge, 1986.
9. Son Tran and Artur Garcez. Logic extraction from deep belief networks. In *ICML 2012 Representation Learning Workshop*, Edinburgh, July 2012.
10. Son N. Tran and Artur d'Avila Garcez. Knowledge extraction from deep belief networks for images. In *IJCAI-2013 Workshop on Neural-Symbolic Learning and Reasoning*, 2013.