Do we really need all these neurons?

Adriana Romero[1] and Carlo Gatta[2]

[1]Dpt. de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Barcelona, Spain.
[2]Centre de Visió per Computador, Campus UAB, Bellaterra, Spain.
adriana.romero@ub.edu
cgatta@cvc.uab.es

Abstract. Restricted Boltzmann Machines (RBMs) are generative neural networks that have received much attention recently. In particular, choosing the appropriate number of hidden units is important as it might hinder their representative power. According to the literature, RBM require numerous hidden units to approximate any distribution properly. In this paper, we present an experiment to determine whether such amount of hidden units is required in a classification context. We then propose an incremental algorithm that trains RBM reusing the previously trained parameters using a trade-off measure to determine the appropriate number of hidden units. Results on the MNIST and OCR letters databases show that using a number of hidden units, which is one order of magnitude smaller than the literature estimate, suffices to achieve similar performance. Moreover, the proposed algorithm allows to estimate the required number of hidden units without the need of training many RBM from scratch.

Keywords: Retricted Boltzmann Machine, hidden units, unsupervised learning, classification

1 Introduction

A Restricted Boltzmann Machine (RBM) [1–3] is a binary stochastic neural network capable of learning internal representation of data being trained in an unsupervised way. It consists of two layers; a visible layer and a hidden layer, both containing binary stochastic units with a bipartite connectivity among them. The binary states of the visible units are represented as a vector v. Likewise, the binary states of the hidden units are represented as a vector h. The model is governed by an energy function that assigns a probability to each possible pair of vectors (v,h). The joint probability of a visible vector v and a hidden vector h is given by

$$p(v,h) = \frac{1}{Z} \exp\left(-\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j \right), \tag{1}$$

where v_i is the binary state of visible unit i, h_j is the binary state of hidden unit j, a_i and b_j are their corresponding biases, w_{ij} the weights among them and Z is a normalization factor [4].

RBMs have been used as generative models for many kinds of data and as parts of deep belief networks [4]. Since they are able to retrieve internal representation of data through the hidden units, they have been used for unsupervised feature extraction. After training a RBM, the extracted features, namely, the states of the hidden nodes, are used for the classification task. RBMs are usually trained by means of the Contrastive Divergence procedure [3], which requires setting the values of a number of meta-parameters. The number of hidden units used in a RBM has a strong influence on the distributions that the RBM is able to learn. Thus, the representational power of a RBM may be hindered if the number of hidden units is not set properly. The value of this parameter is often chosen on the basis of experience or optimized by extensive search [5]. Hinton [4] proposes a recipe for choosing the number of hidden units based on the number of bits needed to describe each data-vector in a good model. Le Roux and Bengio [6] show that a RBM with N_V visible units and N_H hidden units is a universal approximator when $N_H \geq 2^{N_V} + 1$. Montufar and Ay [7] show that $N_H \geq 2^{N_V - 1} - 1$ are sufficient to approximate all distributions and that $N_H \geq \frac{2^{N_V}}{N_V + 1} - 1$ hidden units are required to represent all distributions in $\{0, 1\}^{N_V}$. In [5], the authors give a theoretical basis for selecting the size of an RBM given an error tolerance. The size of the RBM is chosen according to the Kullback-Leibler divergence among the data distribution and the RBM distribution in an unsupervised way. Thus, a large number of parameters and hidden units is used in the literature to train models.

In this paper, we propose an experiment to determine whether the large amount of hidden units used in the literature is really necessary. For this purpose, we introduce a measure that evaluates the performance gain in the classification task with respect to the number of parameters in the model. Then, we describe an algorithm that uses this measure to incrementally train RBMs. The incremental training provides a way to exploit the results of previously trained RBMs and determine the appropriate number of hidden units according to a trade-off measure between the accuracy gain and the number of parameters in the model. Unlike [5], the method that we propose sets the number of hidden units with a supervised target.

The rest of the paper is organized as follows. Section 2 outlines how the accuracy of a RBM varies with the number of hidden nodes and introduces a trade-off measure between the performance gain and the number of parameters of the RBM. Section 3 describes the proposed algorithm to incrementally train RBMs. Section 4 discusses the results obtained in the previous sections. Finally, in Section 5 conclusions are drawn.

2 Number of Hidden Units

In the literature [4, 6, 7], a large number of hidden units is used in order to be able to model data properly. In this section, we propose an experiment to

determine whether such amount of units is required. Our goal is to define the minimum number of hidden nodes required to obtain satisfactory performance in a classification task.

To do so, we train a set of RBMs, each containing a different number of hidden units. The RBMs are trained using the Contrastive Divergence learning procedure [3]. The parameters are set according to [4]: the weights w are initialized to random values drawn from a normal distribution $\mathcal{N}(0, 0.001)$ and the biases a and b are initialized to 0. The experiment is performed on two different datasets: the MNIST dataset [8] and the OCR letter dataset. [1] The MNIST database is a dataset of handwritten digits divided into a training set of 60000 examples and a test set of 10000 examples, both containing a balanced number of examples for each digit class. The OCR letters dataset contains a set of handwritten lower case letters divided into 10 different sets, corresponding to cross-validation folds. Fig. 1 shows some examples from both datasets. Learning

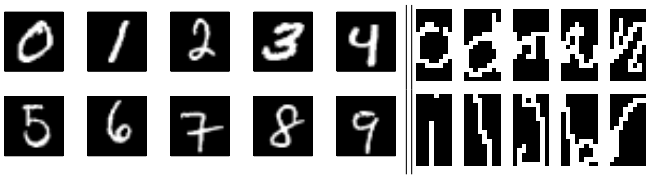


Fig. 1. Examples of MNIST (left) and OCR (right) letters databases.

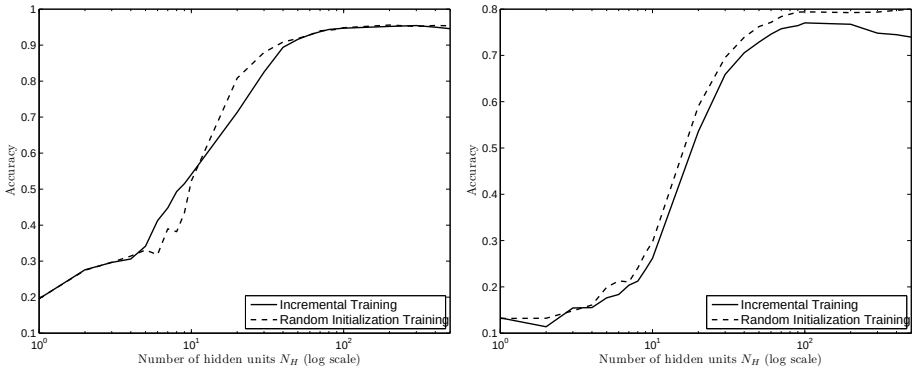
is performed for each RBM with 1-step Contrastive Divergence on the training set[2] with 100 epochs per RBM. Then, each RBM is applied to the test set and, after that, the outputs of the RBM (the hidden layer outputs) are used as inputs in a k-Nearest Neighbor (k-NN) algorithm (with $k = 1$) for the classification task. The performance of each RBM is evaluated in terms of accuracy, as the proportion of examples correctly classified. More complex classifiers can mask the experimental results analysis, so that we keep k-NN as a base classifier.

Fig. 2 (dashed lines) shows the accuracy of applying k-NN classification to the hidden layer of the RBM according to the number of hidden units. Fig. 2(a) shows the results obtained on the MNIST dataset and Fig. 2(b) shows the results obtained on the OCR dataset. As shown in Fig. 2, the accuracy increases rapidly when the number of hidden units is small and converges, in both datasets, as the number of hidden nodes approaches 100. In order to determine the appropriate number of hidden nodes, we propose a measure M that computes the gain in accuracy with respect to the increment in the number of parameters (weights and biases) of the RBM. M is defined as follows

$$M = \frac{\triangle A}{\triangle N_P}, \tag{2}$$

[1] http://ai.stanford.edu/ btaskar/ocr/

[2] Using the code provided in http://www.cs.toronto.edu/ hinton/MatlabForSciencePaper.html



(a) MNIST (b) OCR letters

Fig. 2. Accuracy changing the number of hidden units N_H (log scale) applying a random initialization of the parameters and the incremental training on (a) MNIST dataset and (b) OCR letters database.

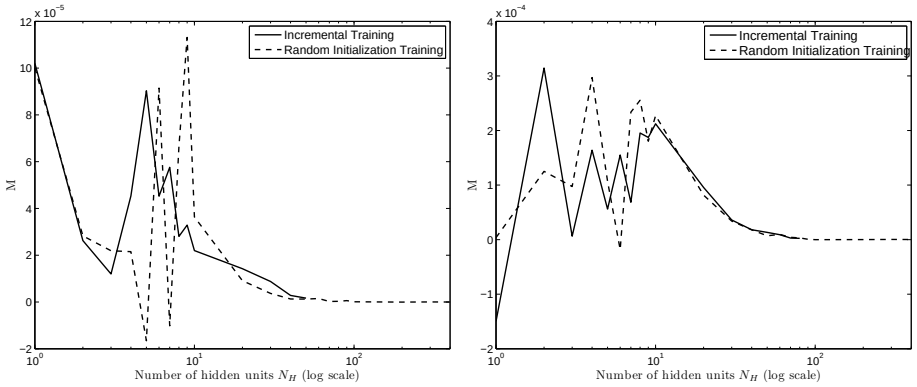
where A is the accuracy and N_P the number of parameters of the RBM, i.e., the sum of the number of weights and the number of hidden biases.

Fig. 3 (dashed lines) shows how M varies as a function of the number of hidden units N_H. As shown in the figure, M oscillates when the number of hidden units is small and stabilizes as the number of hidden nodes in the RBM approaches 100, in both datasets. While the number of nodes is small, adding new nodes to the RBM has a strong impact on the performance of the classification. As we increase the number of hidden units, the gain in performance does not compensate for the increment in the number of parameters. Clearly, when M approaches zero, increasing the number of hidden nodes does not provide a significatively higher accuracy.

3 Incremental Training of RBM

In this section, we describe an algorithm to find the appropriate number of hidden nodes N_H of a RBM thresholding on M (see Eq. (2)), while training the RBM. In Section 2, we trained a set of RBMs setting their initial parameters each time, as explained in [4]. The experiment was performed training RBMs by means one-step Contrastive Divergence with 100 epochs. However, in this section, we aim at incrementally training RBMs until, for a positive accuracy gain, we achieve a value of M close to 0. To do so, RBMs are trained by adding new nodes progressively and reusing the parameters learnt in the previous RBM, i.e., previously trained parameters are maintained while adding new hidden nodes. RBMs are trained as in Section 2, by means of one-step Contrastive Divergence but only with 10 epochs. We expect this training strategy to allow faster convergence than the standard RBM training, where no previous parameter estimation is reused.

Algorithm 1 describes the incremental training procedure. Given a training set D_{train} and a test set D_{test} with their corresponding labels L_{train} and L_{test}



(a) MNIST (b) OCR letters

Fig. 3. M with respect to the number of hidden units N_H (log scale) applying a random initialization of the parameters and the incremental training on (a) MNIST dataset and (b) OCR letters database.

and a threshold thr_M on the measure M, the procedure consists in iteratively adding new hidden units to the RBM until the stopping criterion is satisfied. Let N_V and N_H be the number of visible and hidden units of a RBM, respectively. Let w be the weights connecting both layers of the RBM, a the biases of the visible units and b the biases of the hidden units. The number of biases in a is not affected by the introduction of new hidden units to the RBM, since the number of visible units always remains the same. Therefore, biases a are initialized to 0 (step 3) and maintained after the training of each RBM. The number of weights and hidden biases changes as we increment the number of hidden units. The weights w are initialized randomly and maintained after each training. When adding new hidden nodes, the weights connecting the new hidden node to the visible nodes are set randomly (steps 10-11). Similarly, when a new hidden node is added to the RBM, its corresponding bias is set to 0 and further maintained after each RBM training (step 12). Starting from a RBM with one hidden unit ($N_H = 1$), we increment the RBM maintaining the previously trained parameters and setting the new ones as described in [4], namely the new weights to small random values and the new hidden biases to 0 (steps 9-12). Then, we train the new RBM using one-step Contrastive Divergence (step 13). After that, we classify the data in D_{test} using k-NN algorithm ($k = 1$) on the output of the hidden layer feature space (step 14). Finally, we evaluate the results of the classification in terms of accuracy (step 15) and compute the trade-off M between the increment in accuracy and the number of parameters (step 16-17). We keep adding hidden units and evaluating the performance of the classification until, for a positive accuracy gain, M becomes lower than thr_M (step 19). Given a threshold $thr_M = 2.5 \cdot 10^{-6}$, the incremental training stops at $N_H = 60$ for the MNIST dataset and at $N_H = 90$ for the OCR letters dataset.

Fig. 2 shows how the accuracy of the classification varies when applying the incremental training (solid line) compared to the accuracy of the classification

Algorithm 1 RBM Incremental Training

Require: $D_{train}, L_{train}, D_{test}, L_{test}, thr_M$
Ensure: RBM
$N_V \leftarrow$ input data dimensionality.
$N_H \leftarrow 1$
$a \leftarrow [0...0]$
$w \leftarrow [\,]$
$b \leftarrow [\,]$
$A(0) \leftarrow \frac{1}{\#classes}$
$N_P(0) \leftarrow 0$
repeat
  Increment RBM_{N_H}:
    $w_{new} \leftarrow$ random values from $\mathcal{N}(0, 0.001)$.
    $w \leftarrow [w \;\; w_{new}]$
    $b \leftarrow [b \;\; 0]$
  $RBM_{N_H} \leftarrow$ learn RBM using D_{train}.
  Classify D_{test} using k-NN ($k = 1$) algorithm on RBM_{NH} output feature space.
  $A(N_H) \leftarrow \frac{\#correctly\ \ classified\ \ examples}{\#examples}$
  $N_P(N_H) \leftarrow N_V N_H + N_H$
  $M \leftarrow \frac{A(N_H) - A(N_H - 1)}{N_P(N_H) - N_P(N_H - 1)}$
  $N_H \leftarrow N_H + 1$
until $M < thr_M$ & $A(N_H) - A(N_H - 1) > 0$

when RBM parameters are re-initialized each time (dashed line). As it can be seen in Fig. 2(a) and (b), applying the incremental training provides similar accuracy than applying a regular RBM training, while requiring significantly fewer iterations in the learning procedure. In the experiments of this paper, the training speed up is of about one order of magnitude. Fig. 3 shows how the trade-off M varies when applying the incremental training (solid line) compared to its variations when RBMs are re-initialized at each training (dashed line).

4 Discussion

The experiment proposed in Section 2 aims at determining whether a large amount of hidden nodes, as suggested in the literature [4, 6, 7], is required to obtain a good performance. In [6], the authors claim that at least $2^{N_V} + 1$ hidden nodes are required in order to be able to model any probability distribution. In the case of the experiment on the MNIST database described in Section 2, $N_V = 784$ and, thus, $2^{784} + 1 \approx 10^{236}$ hidden nodes would be required. In [7], the authors demonstrate that $\frac{2^{N_V}}{N_V+1} - 1$ hidden nodes are sufficient. In the case of the MNIST database, that would be at least $\frac{2^{784}}{784+1} - 1 \approx 10^{233}$. In [4], a recipe is provided to estimate this number: "Assuming that the main issue is overfitting rather than the amount of computation at training or test time, estimate how many bits it would take to describe each data-vector if you were using a good model (i.e. estimate the typical negative log_2 probability of a data-vector under a good model). Then multiply that estimate by the number of training cases and use a number of parameters that is about an order of magnitude smaller. If you are using a sparsity target that is very small, you may be able to use more hidden units. If the training cases are highly redundant, as they typically will be for very big training sets, you need to use fewer parameters". However,

determining whether a model is good is not trivial. We consider a model, which assumes pixel independence and compute the information contained in each pixel of the image. To do so, we first binarize the input image so that each pixel in the image has two possible values $\{0, 1\}$. The information of a pixel, considering all the training examples, can be computed as the Shannon Entropy [9]. Since we assume pixel independence, we add the information of each pixel to set the number of bits required to describe each data-vector. We obtain that 298 bits are required to describe the MNIST training set. The proposed model provides a decent representation of the data. However, given that it assumes pixel independence, the model might not be sufficient to describe the data properly. Considering neighboring pixels might improve the model. However, increasing the order of the model, increases the complexity of the model as well. Therefore, defining whether a model is good is a complex task by itself, and out of scope of this paper. We multiply the 298 bits required to describe the data by the number of training cases (60000 in the MNIST database) and use a number of parameters, which is an order of magnitude smaller, as stated in [4]. In this case $298 \cdot 60000 \approx 1.78 \cdot 10^7$, that corresponds to $1.78 \cdot 10^6$ parameters and, thus, 2270 hidden units. The number of hidden units chosen in the basis of experience as in [4] is already significantly smaller than choosing it so that the RBM is a universal approximator ($10^3 \ll 10^{233}$), as expected. Similarly, in the case of the OCR letters dataset, the number of hidden units chosen on the basis of experience would be approximately 10^3, which is significantly smaller than the 10^{36} hidden nodes needed to have a universal approximator. However, our results show that the accuracy of the classification converges as the number of hidden nodes in the RBM approaches 100 in both datasets. Furthermore, the measure M defined in Section 2 provides a way to evaluate if the gain in performance compensates for the increment in the number of parameters of the model. As the number of hidden nodes increases, the performance gain tends to be smaller and might not compensate for the amount of new parameters introduced in the model. Therefore, a RBM with a smaller number of hidden units might be sufficient to obtain a good accuracy.

The algorithm proposed in Section 3 trains RBMs incrementally, reusing the parameters trained in previous smaller RBMs. It is important to notice that the incremental training learns RBMs with 10 epochs, while the standard RBM training needs 100 epochs. Fig. 2 shows the performance of the incremental training compared to the training used in Section 2. As shown in the figure, the incremental training is able to obtain similar accuracy when applied to MNIST dataset, while requiring less epochs in the learning procedure. However, when dealing with the OCR letters database, the standard training outperforms the incremental training for a significant range of N_H. However, after choosing the appropriate number of hidden nodes according to the incremental training algorithm, the performance of the classification could be easily improved by retraining the selected RBM with more epochs.

5 Conclusions

In this paper, we proposed an experiment in order to analyze how the number of hidden nodes affects the performance in a classification task. We defined a measure to determine the minimum number of hidden nodes required to achieve a sufficient performance and we proposed an incremental training algorithm for RBMs that exploits the previously trained parameters to learn new RBMs. Experiments conducted on the MNIST and OCR letters datasets suggest that using significantly fewer hidden units than the proposed in the literature suffices to obtain a good performance. The proposed algorithm was validated on two datasets, showing that it is able to provide similar accuracy to the standard RBM training, while requiring significantly fewer epochs to converge.

Acknowledgements. The work of C. Gatta is supported by the Spanish Ministry of Science and Innovation MICINN under a Ramon y Cajal Research Fellowship. The work of A. Romero is supported by an Ajuts de Personal Investigador en Formació grant of the University of Barcelona. This work has been supported in part by project TIN2009-14404-C02.

References

1. P. Smolensky, "Information processing in dynamical systems: foundations of harmony theory," in *Parallel distributed processing: explorations in the microstructure of cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA, USA: MIT Press, 1986, vol. 1, ch. 6, pp. 194–281.
2. Y. Freund and D. Haussler, "Unsupervised learning of distributions on binary vectors using two layer networks," in *Advances in Neural Information Processing Systems 4*, San Mateo, CA, USA, 1992, pp. 912–919.
3. G. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14(8), pp. 1771–1800, 2002.
4. ——, "A practical guide to training restricted boltzmann machines, version 1," University of Toronto, Tech. Rep., 2010.
5. G. Montufar, J. Rauh, and N. Ay, "Expressive power and approximation errors of restricted boltzmann machines." in *NIPS*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 415–423.
6. N. Le Roux and Y. Bengio, "Representational power of restricted Boltzmann machines and deep belief networks," *Neural Computation*, vol. 20, no. 6, pp. 1631–1649, 2008.
7. G. Montufar and N. Ay, "Refinements of universal approximation results for deep belief networks and restricted boltzmann machines," *Neural Computation*, vol. 23, no. 5, pp. 1306–1319, 2011.
8. Y. Lecun and C. Cortes, "The MNIST database of handwritten digits."
9. C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.