Improving Convergence of Restricted Boltzmann Machines via a Learning Adaptive Step Size

Noel Lopes[1,2] and Bernardete Ribeiro[1,3]

[1] CISUC - Center for Informatics and Systems of University of Coimbra, Portugal
noel@ipg.pt, bribeiro@dei.uc.pt
[2] UDI/IPG - Research Unit, Polytechnic Institute of Guarda, Portugal
[3] Department of Informatics Engineering, University of Coimbra, Portugal

Abstract. Restricted Boltzmann Machines (RBMs) have recently received much attention due to their potential to integrate more complex and deeper architectures. Despite their success, in many applications, training an RBM remains a tricky task. In this paper we present a learning adaptive step size method which accelerates its convergence. The results for the MNIST database demonstrate that the proposed method can drastically reduce the time necessary to achieve a good RBM reconstruction error. Moreover, the technique excels the fixed learning rate configurations, regardless of the momentum term used.

Keywords: Restricted Boltzmann Machines, Deep Belief Networks, Deep learning, Adaptive step size.

1 Introduction

Restricted Boltzmann Machines (RBMs) are becoming increasingly popular due to their unsupervised learning characteristics and inherent ability to cope with missing data [12], but mostly due to their potential to integrate more complex and deeper architectures. Deep architectures have recently gain momentum, raising the interest of machine learning researchers [6] due to theoretical and empirical results suggesting that they can be exponentially more efficient than shallow ones [11]. Much of this interest derived from the development of Deep Belief Networks (DBNs), recently proposed by Hinton et al. [4].

DBNs have been successfully applied to several domains including classification, regression, dimensionality reduction, object segmentation, information retrieval, robotics, natural language processing, and collaborative filtering among others [2]. DBNs are composed of several RBMs stacked on top of each other. The idea is to progressively extract higher-level dependencies from the original input data, thereby improving the ability of the network as a whole to capture the underlying regularities of the data [10].

Each RBMs learns a generative model from the data distribution, using the Contrastive Divergence (CD) algorithm, which is an approximation of the true gradient of the data likelihood, since calculating the later is not feasible [12].

L. Alvarez et al. (Eds.): CIARP 2012, LNCS 7441, pp. 511–518, 2012.

Training a DBNs is often a computationally expensive process that involves training independently several RBMs and may require a considerable amount of time. Moreover the proper choice of the learning rate and momentum parameters is a fundamental aspect of the training procedure that may affect considerably the networks convergence. In particular the learning rate is correlated with the learning speed [12]. In this paper we propose the use of an adaptive step size technique, which solves the difficulty of choosing an adequate learning rate and accelerates the training convergence.

The remainder of this paper is organized as follows. Section 2 details both the RBMs and DBNs. Section 3 describes the proposed adaptive step size technique. Section 4 presents and discusses the results. Finally, section 5 summarizes the paper contributions and addresses directions for future work.

2 Restricted Boltzmann Machines

An RBM is an energy-based generative model that consists of a layer of binary visible units, $\mathbf{v}$, and a layer of binary hidden units, $\mathbf{h}$, connected by symmetrically weighted connections [5], as depicted in Figure 1.

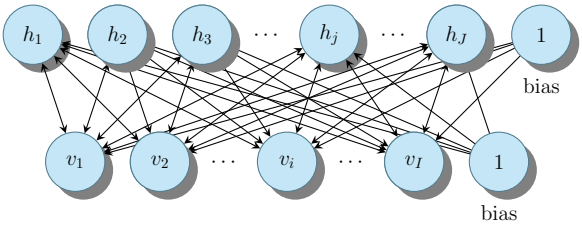

Fig. 1. Schematic representation of a Restricted Boltzmann Machine (RBM)

Given an observed state, the energy of the joint configuration of the visible and hidden units $(\mathbf{v}, \mathbf{h})$ is given by (1):

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{I} a_i v_i - \sum_{j=1}^{J} b_j h_j - \sum_{j=1}^{J}\sum_{i=1}^{I} w_{ji} v_i h_j \ , \tag{1}$$

where a_i represents the bias of the visible unit i, b_j the bias of the hidden unit j and w_{ji} the weight associated to the connection between unit i and unit j.

The RBM assigns a probability for each configuration $(\mathbf{v}, \mathbf{h})$, using the energy function given by (2):

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

where Z is the partition function, obtained by summing the energy of all possible $(\mathbf{v}, \mathbf{h})$ configurations:

$$Z = \sum_{v,h} e^{-E(\mathbf{v},\mathbf{h})} \ . \tag{3}$$

Given a random input configuration $\mathbf{v}$, the state of the hidden unit j is set to 1 with probability:

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_{i=1}^{I} v_i w_{ji}) \ , \tag{4}$$

where $\sigma\,(x)$ is the sigmoid function $\frac{1}{(1+e^{-x})}$. Similarly, given a random hidden vector the state of the visible unit i can be set to 1 with probability:

$$p(v_i = 1|\mathbf{h}) = \sigma(a_i + \sum_{j=1}^{J} h_j w_{ji}) \ . \tag{5}$$

The probability assigned to a visible vector, $\mathbf{v}$, is given by (6):

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}) = \sum_{\mathbf{h}} p(\mathbf{v}|\mathbf{h}) p(\mathbf{h}) \tag{6}$$

Hence, the probability assigned to a specific training vector $\mathbf{v}$ can be raised by adjusting the weights and the biases of the network in order to lower the energy of that particular vector while raising the energy of all the others. To this end, we can performing a stochastic gradient ascent on the log-likelihood surface of the training data, by computing the derivative of the log probability with respect to the weights of the network, which is given by (7):

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_\infty \tag{7}$$

where $\langle \cdot \rangle_0$ denotes the expectations for the data distribution (p_0) and $\langle \cdot \rangle_\infty$ denotes the expectations for the model distribution (p_∞) [7]. Unfortunately, computing $\langle v_i h_j \rangle_\infty$ is intractable as it requires performing alternating Gibbs sampling for a very long time [5]. To solve this problem, Hinton et al. proposed a much faster learning procedure – the Contrastive Divergence (CD-k) algorithm [3,5], whereby $\langle . \rangle_\infty$ is replaced by $\langle \cdot \rangle_k$ for small values of k [7].

Using the specified procedure the following rules can be applied in order to correct the weights and bias of the network:

$$\Delta w_{ij} = \gamma(\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_k) \tag{8}$$

$$\Delta b_j = \gamma(\langle h_j \rangle_0 - \langle h_j \rangle_k) \tag{9}$$

$$\Delta a_j = \gamma(\langle v_i \rangle_0 - \langle v_i \rangle_k) \tag{10}$$

where γ represents the learning rate.

An RBM by itself is limited in what it can represent and its true potential emerges when several RBMs are stacked together to form a DBN [8].

2.1 Deep Belief Networks

RBMs, were recently proposed by Hinton et al. along with an unsupervised greedy learning algorithm for constructing the network one layer at a time [4].

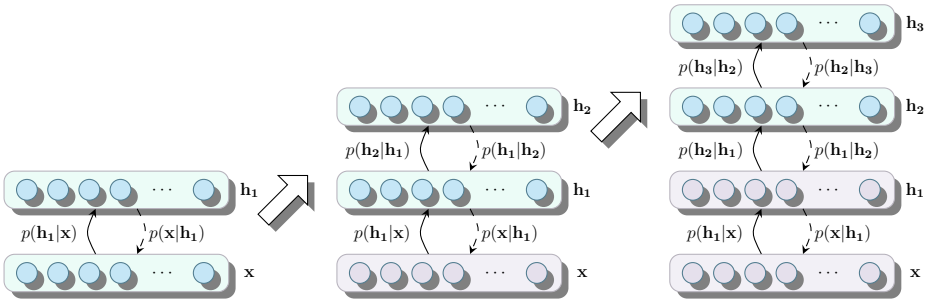

Fig. 2. Training process of a Deep Belief Network (DBN) with one input layer, $\mathbf{x}$, and three hidden layers $\mathbf{h_1}$, $\mathbf{h_2}$, $\mathbf{h_3}$. From left to right, layers with lighter color represent those already trained, while layers with darker color belong to the RBM being trained.

The idea is to train independently each layer using an RBM network that models the output of the previous layer. This approach represents an efficient way of learning an otherwise complicated model, by combining multiple and simpler (RBM) models, learned sequentially [4]. Figure 2 represents this process.

The first layers of the network are expected to extract low-level features from the input data while the upper layers are expected to gradually refine previously learn concepts, therefore producing more abstract concepts [7]. The training process is done in an unsupervised manner allowing the system to learn complex functions by mapping the input to the output directly from data, without depending on human-crafted features [2]. However, the output of the top layer can easily be fed to a conventional supervised classifier [5,10].

Although CD-1 does not provide a very good estimate of the maximum-likelihood, this is not an issue when the features learned serve as inputs for an higher-level RBM [5]. In fact, for RBMs that integrate a DBN, it is not necessarily a good idea to use another form of CD-k that may provide closer approximations to the maximum-likelihood, but does not ensure that the hidden features retain most of the information contained in the input data vectors [5]. This is consistent with the results obtained by Swersky et al. for the MNIST dataset, where the best DBN results were obtained for CD-1 [14].

3 Proposed Approach

Numerous techniques have been proposed for accelerating the convergence of the Back-Propagation (BP) algorithm. An analysis of these can be found in Zainuddin et al. [15]. Among those, the adaptive step size, proposed by Silva and Almeida [13,1], consists of using an individual learning rate (step size) parameter, γ_{ji}, for each weight connection, w_{ji}, instead of a global learning rate for all the network. Adapting this idea for the case of an RBM, at each CD-k iteration, the step sizes are adjusted according to the sign changes:

$$\gamma_{ji} = \begin{cases} u\gamma_{ji}^{old} & \text{if } (\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_k)(\langle v_i h_j \rangle_0^{old} - \langle v_i h_j \rangle_k^{old}) > 0 \\ d\gamma_{ji}^{old} & \text{if } (\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_k)(\langle v_i h_j \rangle_0^{old} - \langle v_i h_j \rangle_k^{old}) < 0 \end{cases} \quad (11)$$

where $u > 1$ (up) represents the increment factor for the step size and $d < 1$ (down) the decrement factor. If two consecutive updates have the same direction the step size of that particular weight is increased. For updates with opposite directions the step size is decreased, thus avoiding oscillations in the learning process due to excessive learning rates. The underlying ideas of this procedure is to find near-optimal step sizes that would allow to bypass ravines on the error surface. Moreover, the technique is especially effective for ravines that are parallel (or almost parallel) to some axis [1].

Additionally, each connection has its own momentum, $\alpha_{ji} = \gamma_{ji}\alpha$, proportional to the global momentum configuration, α, and to the step sizes. Moreover, according to our tests, it is advantageous to clamp α_{ji}, such that $0.1 \leq \alpha_{ji} \leq 0.9$.

4 Results and Discussion

4.1 Experimental Setup

In order to evaluate the performance of the proposed technique, we conducted several tests using the MNIST database of handwritten digits, available at `http://yann.lecun.com/exdb/mnist/`. This database contains 60,000 training samples. However, for the tests conducted in this work we only use the first 1,000 samples. Each sample consists of a 28×28 pixel image of a hand-written digit. Hence, each sample has 784 inputs.

The tests were conducted using our Graphics Processing Units (GPU) implementation of RBMs [9] on a Core 2 Quad CPU Q9300 (2.5 GHz) with an NVIDIA GTX 280.

The adaptive step size technique was compared with three different fixed learning rate settings ($\gamma = 0.1$, $\gamma = 0.4$ and $\gamma = 0.7$), while using three distinct momentum terms ($\alpha = 0.1$, $\alpha = 0.4$ and $\alpha = 0.7$). For adaptive step size technique the initial learning rate of each connection was set to 0.1 and the increment, u and decrement, d, factors were set respectively to 1.2 and 0.8. Altogether, twelve configuration settings were used (three for the adaptive step technique and nine for the fixed learning rates). For statistical significance, we conducted 30 tests per configuration, using an RBM with 784 inputs and 100 outputs. Each test starts with a different set of weights, but for fairness all the configurations use the same weight settings according to the test being performed.

4.2 Benchmark Results

Figure 3 show the evolution of the Root Mean Square Error (RMSE) of the reconstruction, according to to the learning rate, γ, and momentum, α, settings.

Independently of the learning rate used, the best results were obtained for a momentum $\alpha = 0.1$, while the worst solutions were obtained for $\alpha = 0.7$.

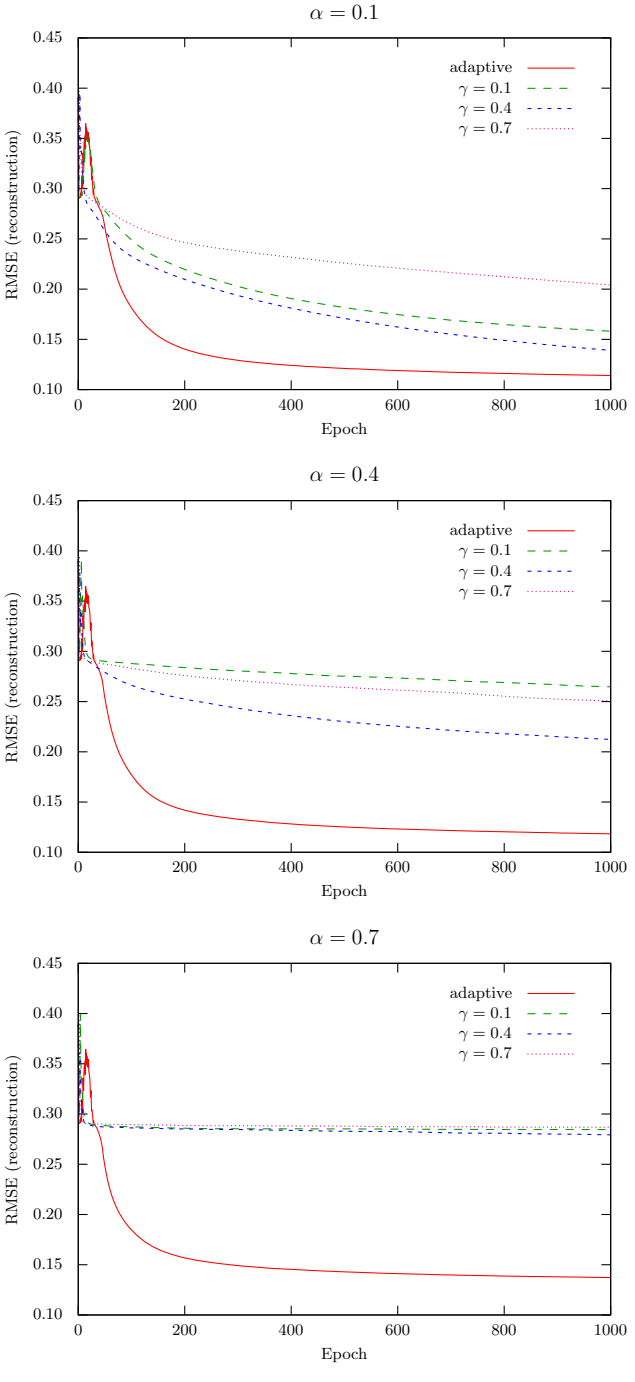

Fig. 3. Average reconstruction RMSE according to the learning parameters

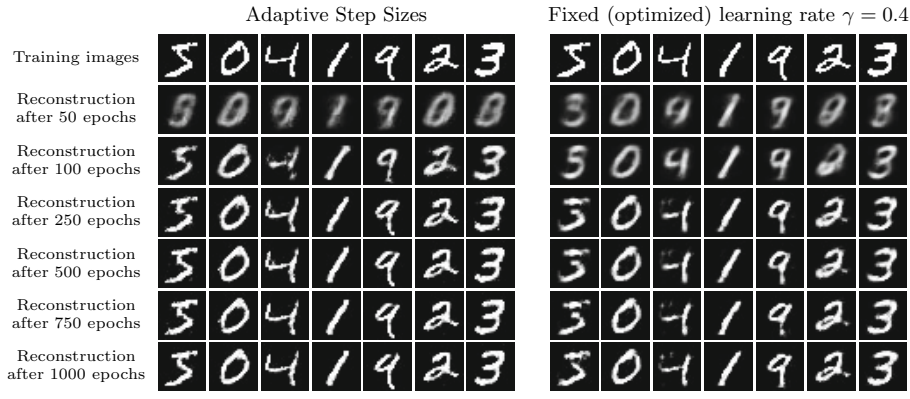

Fig. 4. Impact of the step size technique on the convergence of an RBM ($\alpha = 0.1$)

As expected the adaptive step size technique excels all the fixed learning rate configurations, regardless of the momentum term used. The discrepancy is quite significant (2.51%, 9.39% and 14.20% relatively to the best fixed learning rate solution, respectively for $\alpha = 0.1$, $\alpha = 0.4$ and $\alpha = 0.7$) and demonstrates the usefulness of the proposed technique and its robustness to an inadequate choice of the momentum. Moreover, in order to achieve better results than those obtained after 1000 epochs, using a fixed learning rate, we would only require 207, 68 and 48 epochs, respectively for $\alpha = 0.1$, $\alpha = 0.4$ and $\alpha = 0.7$.

Figure 4 shows the quality of the reconstruction of the original images in the database, for both the best network trained with a fixed learning rate ($\gamma = 0.4$, $\alpha = 0.1$) and the best network trained with the step size technique.

Training a network for 1,000 epochs (in the hardware configuration described earlier) using the adaptive step size takes on average 76.63±0.09 seconds, while training the same network with a fixed learning rate takes 76.12±0.05 seconds. Thus the overhead of using this method is not significant, while the convergence of the network is considerably enhanced.

Additionally, by using the adaptive step size technique, we are no longer required to search for a suitable γ parameter. Moreover, the step size method can easily recover from a bad choice of the initial learning rate [1] and the parameters u and d are easily tuned (the values chosen for this problem will most likely yield good results for other problems).

5 Conclusions and Future Work

The CD-k algorithm performs an efficient gradient ascent approximation. Nevertheless training an RBM is a difficult task. Experimental results suggest that finding a suitable learning rate is fundamental to successfully accomplish this task. In this paper we presented an adaptive step size method that solves this

problem by using a different learning rate and momentum for each connection. At each iteration the technique seeks to find the near-optimal step sizes in order to improve convergence. The proposed method yielded excellent results in the MNIST handwritten, reducing drastically the reconstruction error. Moreover, the running overhead in the GPU is imperceptible.

Future work will focus on applying the proposed technique to real-world problems.

References

1. Almeida, L.B.: C1.2 Multilayer perceptrons. In: Handbook of Neural Computation, pp. C1.2:1–C1.2:30. IOP Publishing Ltd. and Oxford University Press (1997)
2. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 2(1), 1–127 (2009)
3. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Comput. 14, 1771–1800 (2002)
4. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. 18, 1527–1554 (2006)
5. Hinton, G.: A practical guide to training restricted boltzmann machines. Tech. rep., Dep. of Computer Science, University of Toronto (2010)
6. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Proc. of the 24th Intl. Conference on Machine Learning, pp. 473–480. ACM (2007)
7. Le Roux, N., Bengio, Y.: Representational power of restricted boltzmann machines and deep belief networks. Neural Comput. 20, 1631–1649 (2008)
8. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proc. of the 26th Annual Intl. Conference on Machine Learning, pp. 609–616. ACM (2009)
9. Lopes, N., Ribeiro, B.: Restricted boltzmann machines and deep belief networks on multi-core processors. In: IEEE World Congress on Computational Intelligence (2012)
10. Ranzato, M., Boureau, Y., LeCun, Y.: Sparse feature learning for deep belief networks. In: Advances in Neural Information Processing Systems, vol. 20 (2007)
11. Roux, N.L., Bengio, Y.: Deep belief networks are compact universal approximators. Neural Computation 22(8), 2192–2207 (2010)
12. Schulz, H., Müller, A., Behnke, S.: Investigating convergence of restricted boltzmann machine learning. In: NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning, Whistler, Canada (2010)
13. Silva, F.M., Almeida, L.B.: Acceleration Techniques for the Backpropagation Algorithm. In: Almeida, L.B., Wellekens, C. (eds.) EURASIP 1990. LNCS, vol. 412, pp. 110–119. Springer, Heidelberg (1990)
14. Swersky, K., Chen, B., Marlin, B., de Freitas, N.: A tutorial on stochastic approximation algorithms for training restricted boltzmann machines and deep belief nets. In: Information Theory and Applications Workshop (2010)
15. Zainuddin, Z., Mahat, N., Hassan, Y.A.: Improving the convergence of the backpropagation algorithm using local adaptive techniques. Intl. Journal of Computational Intelligence, 172–175 (2005)