Unsupervised Learning and Log-likelihood Analysis with Restricted Boltzmann Machines

Elias Maria Bonasera, Alberto Casellato, Nicola Garbin, and Francesco Pazzocco
(Dated: March 31, 2025)

We present a comprehensive study on the use of restricted Boltzmann machines (RBMs) for unsupervised learning of handwritten digit features from the MNIST dataset. Through experiments with a custom Python implementation, we focus on the RBM architecture, the training via Contrastive Divergence, and extensive hyperparameter tuning. We detail the experimental setup, the log-likelihood estimation method, and evaluate different configurations. The results highlight both the strengths and limitations of the approach, suggesting directions for future improvements.

INTRODUCTION

Unsupervised learning is essential for extracting features from unlabeled data, and probabilistic graphical models (PGMs), such as Markov Random Fields (MRFs), provide a framework for this task. Restricted Boltzmann Machines (RBMs) are a specific type of MRF with a bipartite structure (a visible and a hidden layer with no intra-layer connections). Their ability to model complex distributions and learn efficient representations of intricate data relatively fast makes them interesting both theoretically and for practical applications [1–3].

In this work, we present a complete implementation of a Restricted Boltzmann Machine (RBM) and analyze its behavior through systematic experiments. The paper is organized as follows: Section 1 presents the theoretical foundation of RBMs, Section 2 details the experimental methodology and implementation specifics, Section 3 shows the experimental results accompanied by graphical analysis, and Section 4 discusses our findings, draws conclusions, and suggests avenues for future research. Supplementary equations and technical details are provided in the Appendix.

1. THEORY

Restricted Boltzmann Machines RBMs are generative models consisting of a visible layer, representing the input data, and a hidden layer that captures latent features.

Mathematically, RMBs consist in a set of D binary visible units i of state $\{v_i\}_{i=1,\dots,D}$ symmetrically connected to a set of L binary hidden units μ of state $\{h_\mu\}_{\mu=1,\dots,L}$. The continuos weight $w_{i\mu}$ quantifies the strength between unit i and unit μ (see Figure **??**). [1, 2].

The energy of a configuration $(\mathbf{v}, \mathbf{h})$ is defined as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_\mu b_\mu h_\mu - \sum_{i,\mu} v_i w_{i\mu} h_\mu, \qquad (1)$$

where a_i and b_μ are the biases for the visible and hidden units, respectively, and $w_{i\mu}$ denotes the weight between them. No intra-layer connections exist, which simplifies the energy function and learning dynamics.

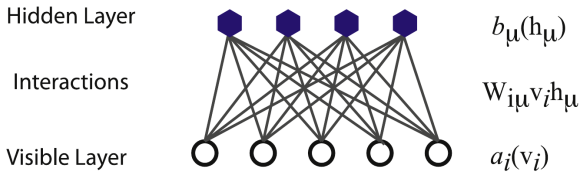

FIG. 1. Bipartite structure of a Restricted Boltzmann Machine. Visible units (bottom) are connected to hidden units (top) with no intra-layer connections.

The probability of a hidden unit being active is computed using the logistic sigmoid for both visibile and hidden units:

$$p(h_\mu = 1|\mathbf{v}) = \sigma\left(b_\mu + \sum_i v_i w_{i\mu}\right), \qquad (2)$$

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_\mu h_\mu w_{i\mu}\right). \qquad (3)$$

The training consists in the minimization of the energy's $\boldsymbol{\theta}$ parameters; and is typically performed via Contrastive Divergence algorithm, which approximates the gradient of the log-likelihood by alternating Gibbs sampling between the visible and hidden layers [1–3].

2. METHODOLOGY

Experimental Setup and Preprocessing The MNIST dataset is used, consisting of 60,000 training images and 10,000 handwritten digit test images. Each image is flattened into a 784-dimensional vector and binarized by thresholding at 0.5. Two encoding schemes are implemented:

- **Bernoulli encoding:** Pixel values in $\{0, 1\}$.
- **Spin encoding:** Pixel values in $\{-1, +1\}$.

The RBM is implemented in a Python class (`RBM`) with configurable hyperparameters such as the number of hidden units, learning rate, optimizer type (SGD or RMSprop), regularization strength, and choice of encoding.

Log-Likelihood Estimation and Parameter Tuning Performance is evaluated using the log-likelihood metric, computed as:

$$\mathcal{L} = \frac{1}{M}\sum_{m=1}^{M} \log\left(\sum_{\mathbf{h}} e^{-E(\mathbf{v}^{(m)},\mathbf{h})}\right) - \log Z, \qquad (4)$$

where Z is the partition function. Due to the computational difficulty of calculating Z, we employ approximations; for the mathematical analysis, see **??**

Hyperparameters varied in our experiments include:

- Number of hidden units (e.g., 3 to 13).
- Number of Contrastive Divergence steps ($k = 1, 2, 3$).
- Learning rate and regularization coefficients.
- Optimizer type (SGD versus RMSprop).
- Data encoding: Bernoulli vs. Spin.

For each configuration, training is repeated over multiple runs (with different random seeds) to ensure statistical significance.

3. RESULTS

Experimental results are presented using several figures. For instance, Figure 2 illustrates the evolution of the log-likelihood over training epochs, showing convergence after approximately 150 epochs. Figure 3 compares the effects of different Contrastive Divergence steps. Additional graphs (e.g., Figure 4) display the impact of varying the number of hidden units. In general, we observe:

- Steady improvement of log-likelihood with training, saturating after a sufficient number of epochs.
- Marginal gains when increasing the number of CD steps, with CD-1 already offering a reasonable trade-off.
- Increased hidden units enhance performance up to a point, beyond which improvements plateau.
- One-hot hidden activations (POTS) and spin encoding can lead to instability or degraded performance.

4. DISCUSSION AND CONCLUSION

Our experiments confirm that RBMs are capable of learning useful representations of handwritten digits in an unsupervised fashion. The analysis shows that:

- Proper initialization and regularization are critical for stable training.

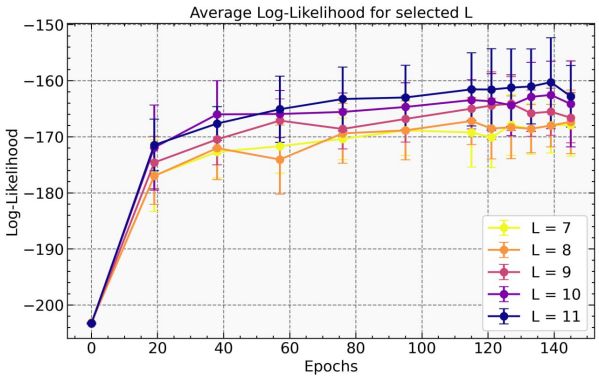

FIG. 2. Evolution of the log-likelihood over training epochs. Results are averaged over multiple runs.

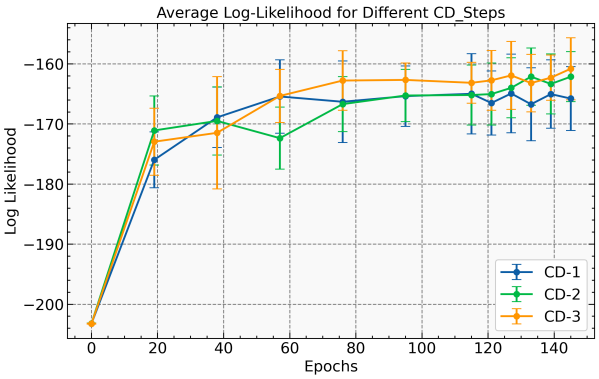

FIG. 3. Log-likelihood evolution for different numbers of Contrastive Divergence steps.

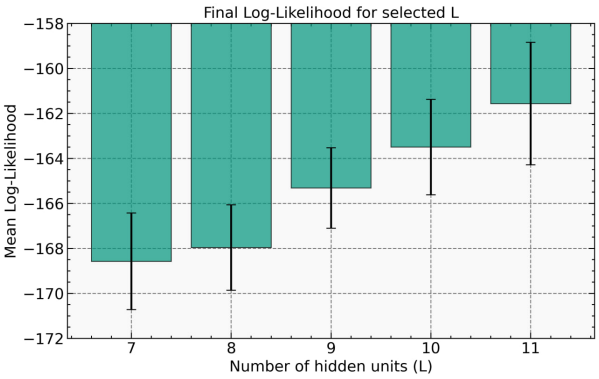

FIG. 4. Bar chart displaying the weighted mean and standard deviation of the log-likelihood over the last epochs for different values of L.

- While increasing hidden units improves performance, the benefit saturates beyond a certain network capacity.
- The choice of optimizer and data encoding significantly affects convergence dynamics.

These findings provide a baseline for further research, which might include adaptive scheduling of CD steps,

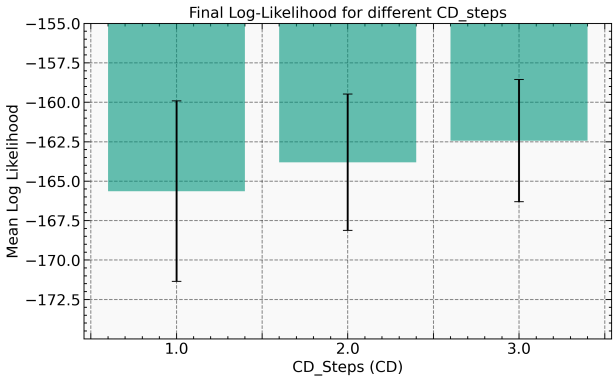

FIG. 5. Bar chart displaying the weighted mean and standard deviation of the log-likelihood over the last epochs for different numbers of CD steps.

deeper architectures (e.g., stacked RBMs), or alternative estimation techniques (such as AIS). Overall, the work not only demonstrates the practical viability of RBMs but also highlights their limitations and areas for future improvement.

5. APPENDIX

Gradient Update Equations and Regularization

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \cdot \eta_t \cdot \text{sign}(\boldsymbol{\theta}), \tag{5}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t \pm \eta_t \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}), \tag{6}$$

$$\begin{cases} \boldsymbol{g}_t = \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}), \\ \boldsymbol{s}_t = \beta \boldsymbol{s}_{t-1} + (1-\beta)\boldsymbol{g}_t^2, \\ \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t \pm \eta_t \frac{\boldsymbol{g}_t}{\sqrt{\boldsymbol{s}_t + \epsilon}}. \end{cases} \tag{7}$$

Log-likelihood and Partition Function Approximations $\mathcal{L}$ of the data:

$$\mathcal{L} = \frac{1}{M} \sum_{m=1}^{M} \ell_\theta(v^{(m)}) \tag{8}$$

$$\ell_\theta(v^{(m)}) = \log \sum_h e^{-E(v,h)} - \log Z \tag{9}$$

where M is the number of data points, Z is the partition function, $E(v,h)$ in the exponential argument $e^{-E(v,h)} = G(h)\prod_i e^{H_i(h)v_i}$ is the energy function, with $G(h) = \prod_\mu e^{b_\mu h_\mu}$ and $H_i(h) = a_i + \sum_\mu w_{i\mu} h_\mu$, and, in Eq. 9, the index h of the summation represents each possible state of the hidden layer.

Z approximations (including second-order Taylor expansion) and normalization techniques (e.g., using a scaling factor in the Bernoulli case or approximating $2\cosh(H_i)$ for spin encoding) are greatly discussed in [4] here we report the final Z. For Bernoulli units, the partition function is manipulated as:

$$Z = q^D \sum_{\mathbf{h}} G(\mathbf{h}) \prod_{i=1}^{D} \frac{1 + e^{H_i(\mathbf{h})}}{q}, \tag{10}$$

and for spin encoding:

$$Z = \sum_{\mathbf{h}} G(\mathbf{h}) \prod_{i=1}^{D} 2\cosh\left(H_i(\mathbf{h})\right). \tag{11}$$

[1] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, Physics Reports **810**, 1 (2019).

[2] M. Bortoletto, *Study of performances for Restricted Boltzmann Machines*, Final dissertation, Università degli Studi di Padova, Dipartimento di Fisica e Astronomia "Galileo Galilei" (2021).

[3] G. E. Hinton, in *Neural Networks: Tricks of the Trade: Second Edition*, edited by G. Montavon, G. B. Orr, and K.-R. Müller (Springer, 2012) pp. 599–619.

[4] M. Baiesi, "Log-likelihood computation for restricted boltzmann machines (rbms)," (2025), unpublished manuscript.