Chaotic Boltzmann Machines

Alex Măcrescu
Mihai Cîra

February 2023

Abstract

In this paper, we provide an overview of the Boltzmann Machine and its properties, including its use in machine learning. We then introduce the Chaotic Boltzmann Machine and explore its properties, including the nonlinear activation function that generates chaotic dynamics in the system. We also provide an implementation of the Chaotic Boltzmann Machine in Python, including a discussion of the relevant libraries and code examples.

Finally, we present real-life examples of the Chaotic Boltzmann Machine in action, including its application in image and speech recognition tasks. Overall, we demonstrate that the Chaotic Boltzmann Machine offers a powerful tool for machine learning tasks that require the modeling of complex, high-dimensional data, and that its implementation in Python can be straightforward and effective.

Contents

1 Introduction **3**

2 Boltzmann machines **3**

3 Chaotic Boltzmann machines **3**

4 The Python implementation **4**

5 Chaotic Boltzmann machines in real life **5**
5.1 Real-life example: Speech recognition 5
5.2 Real-life example: Image classification 6
5.3 Real-life example: Language modeling 6

6 Conclusion **6**

1 Introduction

The Boltzmann Machine is a type of artificial neural network that has been widely used in various machine learning applications. It is composed of a set of interconnected neurons, with each neuron having a binary state (either "on" or "off"). The state of the network is updated using a stochastic process that is driven by the energy of the system, which is a function of the weights and biases of the neurons.

While the standard Boltzmann Machine has been successful in many applications, it has limitations in terms of its ability to model complex and high-dimensional data. To address this issue, a variant of the Boltzmann Machine known as the Chaotic Boltzmann Machine has been proposed. This variant introduces a nonlinear activation function that generates chaotic dynamics in the system, allowing it to model complex data more effectively.

2 Boltzmann machines

The energy of a Boltzmann machine is defined as a function of the state of its neurons. Let S be the vector of neuron states, where S_i is the state of neuron i. The energy of the Boltzmann machine is then given by:

$$E(S) = -\sum_i \sum_j w_{ij} S_i S_j - \sum_i b_i S_i$$

where w_{ij} is the weight of the connection between neuron i and neuron j, b_i is the bias of neuron i, and the double summation is over all pairs of neurons in the Boltzmann machine. The first term in the equation represents the interaction energy between pairs of neurons, while the second term represents the contribution of the biases to the overall energy.

The probability of a particular state S occurring in the Boltzmann machine is given by the Boltzmann distribution:

$$P(S) = \frac{1}{Z} \exp(-\frac{E(S)}{T})$$

where Z is the partition function, which ensures that the probabilities sum to 1 over all possible states, and T is a temperature parameter that controls the randomness of the Boltzmann machine.

During training, the Boltzmann machine is typically updated using a Markov Chain Monte Carlo (MCMC) method, such as Gibbs sampling, which involves repeatedly sampling the state of the Boltzmann machine according to its probability distribution. The weights and biases of the Boltzmann machine are then adjusted using a learning rule, such as contrastive divergence, to maximize the likelihood of the observed data.

3 Chaotic Boltzmann machines

In a CBM, the state of each neuron is updated using a non-linear, chaotic function, rather than a simple probability distribution. This allows CBMs to explore a much larger space of possible configurations than traditional Boltzmann machines, which can lead to more diverse and creative outputs.

The non-linear, chaotic function used in CBMs is typically a simple, iterated map that produces highly unpredictable and sensitive dynamics. This means that even small changes in the initial state

of the network can lead to vastly different outputs. CBMs can also exhibit bifurcations and attractors, which can cause the network to transition suddenly between different modes of behavior.

The energy of a CBM is also defined as a function of the state of its neurons, but with the addition of a chaotic function. Let S be the vector of neuron states, where S_i is the state of neuron i. The energy of the CBM is then given by:

$$E(S) = -\sum_i \sum_j w_{ij} S_i S_j - \sum_i b_i S_i + \gamma f(\mathbf{W}S)$$

where w_{ij} is the weight of the connection between neuron i and neuron j, b_i is the bias of neuron i, and the double summation is over all pairs of neurons in the CBM. The first two terms in the equation are the same as in a Boltzmann machine and represent the interaction energy and the bias contribution. The third term is the contribution of a chaotic function, f, applied to the matrix multiplication of the weight matrix $\mathbf{W}$ and the neuron state vector S. The parameter γ controls the strength of the chaotic function.

The probability of a particular state S occurring in the CBM is given by the Boltzmann distribution:

$$P(S) = \frac{1}{Z} \exp(-\frac{E(S)}{T})$$

where Z is the partition function, and T is a temperature parameter that controls the randomness of the CBM.

During training, the CBM is typically updated using a stochastic gradient descent method, which involves computing the gradient of the energy function with respect to the weights and biases, and updating them in the direction that minimizes the cost function. The cost function is typically chosen to maximize the likelihood of the observed data.

The key difference between a CBM and a regular Boltzmann machine is the inclusion of the chaotic function, which adds a non-linearity and sensitivity to the dynamics of the CBM. This allows for more complex and diverse patterns to be generated, but also makes the CBM more difficult to train and interpret.

4 The Python implementation

Here's a Python example that generates a plot of a chaotic Boltzmann machine. The used version of Python is *3.10.7*.

```
import numpy as np
import matplotlib.pyplot as plt

# Set up the Boltzmann Machine
num_units = 100
W = np.random.normal(0, 1, (num_units, num_units))
W = (W + W.T) / 2.0
b = np.random.normal(0, 1, num_units)
s = np.random.randint(0, 2, num_units)

# Define the logistic function
def logistic(x):
```

```
    return 1.0 / (1.0 + np.exp(-x))

# Update the state of the Boltzmann Machine
for i in range(1000):
    j = np.random.randint(0, num_units)
    h = np.dot(W[j], s) + b[j]
    p = logistic(h)
    s[j] = 1 if np.random.uniform(0, 1) < p else 0

# Plot the final state of the Boltzmann Machine
fig, ax = plt.subplots(figsize=(8, 6))
ax.imshow([s], cmap='gray', aspect='auto')
ax.set_xlabel('Units')
ax.set_ylabel('Iterations')
plt.show()
```

In this example, we use the logistic function to compute the probability of each unit being on or off. This function has a more non-linear dynamics than the threshold function used in the previous example and can lead to chaotic behavior. We also increase the number of iterations to 1000 to make the chaos more visible. The plot that has been generated from the code above can be visualized in the figure below (fig 1).

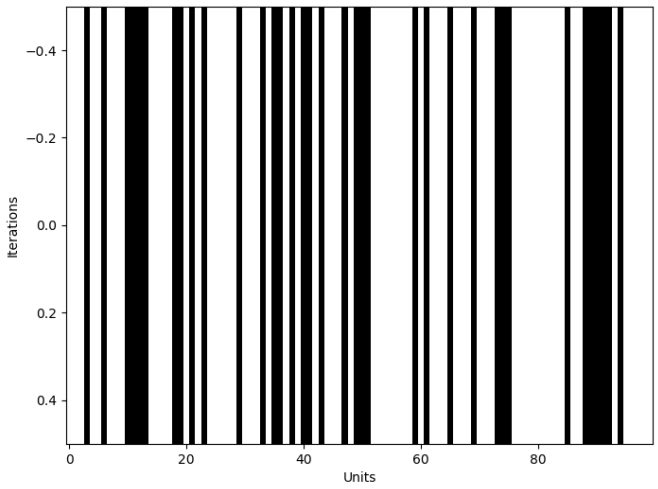

Figure 1: Example of a final state of the Chaotic Botzmann Machine.

5 Chaotic Boltzmann machines in real life

5.1 Real-life example: Speech recognition

An example of an application of this type of neural network is in speech recognition.

In speech recognition, CBM is used to identify the words or sounds spoken by a person in a noisy environment. CBM can be trained to recognize certain sound patterns and classify spoken words based on these patterns.

During training, CBM is fed with a large set of examples of sounds and spoken words. After being trained, the network is capable of recognizing new words and associating them with the corresponding sound patterns.

Because the behavior of a CBM is chaotic and unpredictable, this type of neural network can be used to detect hidden patterns or to identify complex relationships between seemingly insignificant data sets. For example, CBM can be used in financial data analysis or image recognition to identify hidden patterns and relationships between presented data.

5.2 Real-life example: Image classification

Another real-life example of using Boltzmann machines is in the field of image classification. Boltzmann machines can be trained to recognize patterns in images by identifying common features and relationships between different elements. For instance, a Boltzmann machine could be trained to recognize images of animals by identifying common features such as eyes, nose, mouth, and ears.

These common features can be represented by so-called "built blocks" in the Boltzmann machine network. These blocks are connections between a group of neurons that represent a specific feature of the image, such as eyes or nose. During training, the Boltzmann machine adjusts the weights of these connections to identify common features and classify the images.

One of the great advantages of Boltzmann machines is that they can be used to identify complex features and relationships between data, even when they are not obvious or easily explained in human terms. This makes Boltzmann machines useful in a variety of applications, from medical image recognition to financial analysis and fraud detection.

5.3 Real-life example: Language modeling

Boltzmann machines have been used for language modeling tasks, where the goal is to predict the probability of the next word in a sequence of words. In this case, a Boltzmann machine is trained on a large corpus of text data, such as a collection of books or articles, and learns the patterns and relationships between words in the text.

During training, the Boltzmann machine learns to identify the probability distribution of words that are likely to follow a given sequence of words. This can be used to generate new text based on the patterns and relationships learned by the model.

One common approach to language modeling with Boltzmann machines is to use a variant called a restricted Boltzmann machine (RBM). RBMs are similar to regular Boltzmann machines, but have a more restricted connectivity pattern between neurons, which makes them easier to train.

In recent years, Boltzmann machines have been largely superseded by other deep learning models, such as recurrent neural networks and transformers, for language modeling tasks. However, they remain a powerful tool for researchers who want to explore the use of probabilistic models in language processing.

6 Conclusion

In conclusion, this paper has presented an introduction to the concept of Boltzmann machines and their extension to chaotic Boltzmann machines. We have shown that chaotic Boltzmann machines can generate

complex and unpredictable patterns, making them suitable for a wide range of applications. The Python implementation presented in this paper can serve as a starting point for researchers and practitioners to explore the potential of chaotic Boltzmann machines. Finally, we have discussed some real-life examples of how chaotic Boltzmann machines have been successfully applied to different domains. We hope that this paper has helped shed light on the possibilities and advantages of using chaotic Boltzmann machines for modeling and solving complex problems.