Firefly Algorithm for Hyperparameter Tuning of CBOW Word2Vec Embeddings Models

Moussa JAMOR
National school For Computer
Science Artificial intelligence major
ENSIAS Rabat, Morocco
moussa_jamor@um5.ac.ma

***Abstract*— This paper presents the application of a swarm intelligence algorithm for hyperparameter tuning in deep learning. Specifically, we use the Firefly Algorithm to optimize hyperparameters for the Continuous Bag of Words (CBOW) Word2vec Embedding Model.The implementation details and source code can be found at https://github.com/JamorMoussa/FireFly-Optimizer-Deep-Learning.**

I. Introduction

A. Background and Motivation

Hyperparameter tuning is a critical step in the development of deep learning models, significantly influencing their performance. The Continuous Bag of Words (CBOW) model [1], a type of Word2vec embedding, is widely used in natural language processing (NLP) tasks to represent words in a continuous vector space. However, finding the optimal set of hyperparameters for the CBOW model is a challenging and time-consuming process that often relies on trial and error.

B. Swarm Intelligence Algorithms

Swarm intelligence algorithms, inspired by the collective behavior of social creatures like birds, bees, and fireflies, have shown great promise in solving complex optimization problems. The Firefly Algorithm, in particular, mimics the flashing behavior of fireflies to find optimal solutions through iterative search and adjustment.

C. Problem Statement

Manual tuning of hyperparameters in CBOW models can be inefficient and may not always yield the best performance. Traditional methods such as grid search or random search often require extensive computational resources and time. There is a need for more efficient and automated methods to optimize hyperparameters in CBOW models.

D. Proposed Solution

In this paper, we propose the use of the Firefly Algorithm for hyperparameter tuning in CBOW Word2vec embedding models. By leveraging the exploratory capabilities of the Firefly Algorithm, we aim to identify optimal hyperparameter settings more efficiently than traditional methods.

E. Objectives and Contributions

The primary objectives of this paper are to:

- Demonstrate the application of the Firefly Algorithm for hyperparameter tuning in CBOW models.
- Evaluate the performance of the optimized CBOW model against traditional tuning methods.
- Provide insights into the effectiveness of swarm intelligence algorithms in deep learning optimization tasks.

F. Paper Organization

The rest of this paper is organized as follows: Section 2 reviews related work in hyperparameter tuning and swarm intelligence algorithms. Section 3 details the methodology, including the Firefly Algorithm and its application to CBOW. Section 4 presents the experimental setup and results. Section 5 discusses the findings and their implications. Finally, Section 6 concludes the paper.

II. Methodology

A. Overview of the CBOW Model

In their seminal work, Mikolov et al. [1] proposed two model architectures for computing vector representations of words from a large corpus of text, collectively known as Word2vec models. One of these architectures is the Continuous Bag of Words (CBOW). This model is widely used in various NLP tasks, such as sentiment analysis, due to its effectiveness in capturing semantic relationships between words.

The CBOW model operates on a simple yet powerful idea: given a large corpus of text, a neural network is trained to predict a target word based on its surrounding context words within a specified window size W. For instance, with a window size of two, the model uses the two words preceding and the two words following the target word to predict the target word itself. This approach can be considered an unsupervised learning task, as it leverages the natural co-occurrence patterns in the text without requiring labeled data.

Figure 1 represents the model's architecture used for the experiment results. It's based on CBOW proposed in [1]. We start with W window words that are passed to

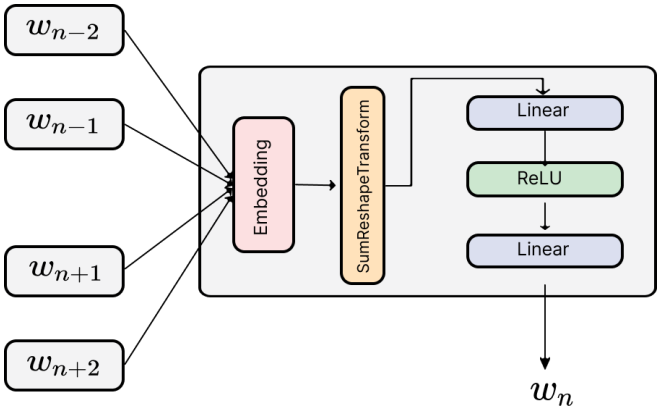

Fig. 1: CBOW Model architecture

an **Embedding** layer, which simply represents a matrix where the number of rows is the vocabulary size and the number of columns is the embedding dimension. Then, it is passed to a **SumReshapeTransform**, which sums the results as proposed in the CBOW architecture [1], but a reshape transformation is added to ensure dimension compatibility. Lastly, we have a **Fully connected layer**, with two **Linear** layers and **ReLU** activation functions. Finally, we have the prediction of the w_n word.

B. Hyperparameters

The performance of the CBOW model is significantly influenced by several hyperparameters. One crucial hyperparameter is the context window size W, which determines the number of words considered on either side of the target word, affecting the capture of semantic relationships.

Another important hyperparameter is the embedding dimension. Each word is represented by a continuous vector, and selecting an appropriate dimension is vital for balancing semantic richness and computational efficiency.

Additional hyperparameters related to the neural network architecture of the CBOW model include:

- **Learning Rate**: The step size used by the optimizer to update model parameters, crucial for convergence.
- **Adam Optimizer Hyperparameters**: β_1 and β_2 control the exponential decay rates for the first and second moments of the gradients, respectively, influencing the behavior of the optimizer during training and affecting convergence and generalization performance.

Choosing the right combination of these hyperparameters is essential for optimizing the CBOW model's performance.

C. Firefly Algorithm

The Firefly Algorithm (FA), introduced by Xin-She Yang [2], is inspired by the flashing behavior of fireflies. Yang argues that FA outperforms other meta-heuristic algorithms, such as Particle Swarm Optimization (PSO), particularly in solving multimodal optimization problems.

In the original paper [2], the Firefly Algorithm is based on three key hypotheses:

- **All-Attractiveness**: All fireflies are attracted to each other regardless of their brightness.
- **Brightness-Based Attraction**: Attractiveness is proportional to brightness, meaning less bright fireflies move towards brighter ones.
- **Random Movement**: If no brighter fireflies are nearby, a firefly moves randomly.

In the Firefly Algorithm, we consider the position of a firefly as a solution vector to an optimization problem. The movement of a firefly is equivalent to the search in the decision space to find the optimal solution x^*. The FA is an iterative algorithm, and we need some time to converge to x^*. We consider the time parameter t, initially starting with $t = 0$. Then, we apply the following update rule, which is the main equation for the firefly algorithm:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} \left(x_i^t - x_j^t\right) + \alpha \boldsymbol{\epsilon}_i^t \tag{1}$$

where r_{ij} denotes the Cartesian distance between firefly i and firefly j, given by

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=0}^{n} \left(x_{i,k} - x_{j,k}\right)^2}$$

Algorithm 1 FireFly Algorithm

Require: Light absorption coefficient γ, maximum generation $MaxGeneration$
Require: Initial population of fireflies x_i $(i = 1, 2, ..., n)$
Generate initial population of fireflies x_i $(i = 1, 2, ..., n)$
Evaluate light intensity I_i at x_i using objective function $f(x)$
Initialize iteration counter $t = 0$
while $t < MaxGeneration$ **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
      if $I_j > I_i$ **then**
        Move firefly i towards j in d-dimension
      end if
      Attractiveness varies with distance r via $\exp[-\gamma r^2]$
      Evaluate new solutions and update light intensity
    end for
  end for
  Rank the fireflies and find the current best
  $t = t + 1$
end while
Post-process results and visualization

D. Hyperparameter Tuning Workflow

Figure 2 illustrates the workflow of hyperparameter tuning using the Firefly Algorithm. The process begins with the

Firefly Algorithm, where a fitness function evaluates the hyperparameters and returns the loss. The goal is to find the best hyperparameters that minimize the loss.

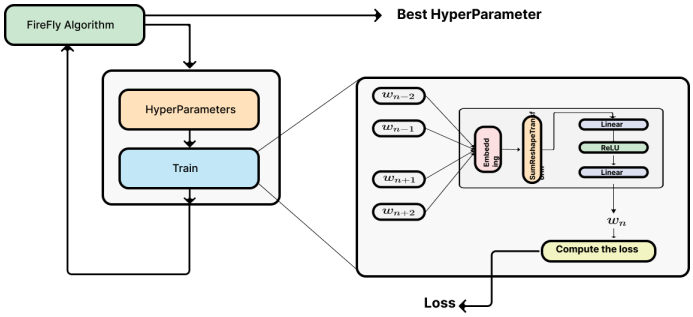

Fig. 2: Hyperparameter Tuning Workflow

III. Experiments and Results

This section presents the experimental results we obtained.

A. *Firefly HyperParameter Tuning*

The first step is to start the Firefly Algorithm and provide a fitness function that accepts hyperparameters, trains the model, and returns the loss function, which is the cross-entropy loss.

The following figures represent the process of running the Firefly Algorithm to determine the optimal hyperparameters:

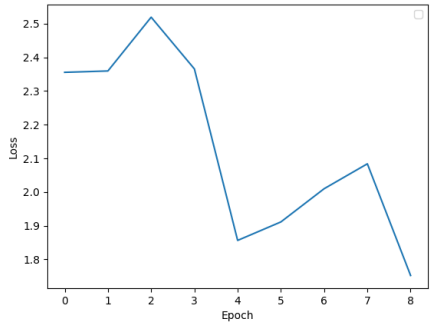

Fig. 3: Figure 1

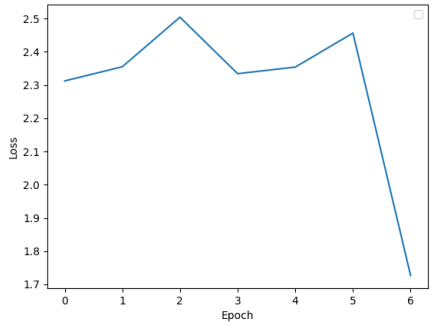

Fig. 4: Figure 2

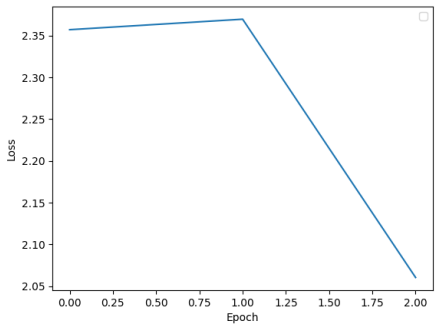

Fig. 5: Figure 3

Parameter	Figure 1	Figure 2	Figure 3
Learning Rate (lr)	0.0093	0.0100	0.0100
β_1	0.899	0.9003	0.901
β_2	0.9981	0.9989	0.9991
Embedding Dimension	2	4	5
Window Size (w)	3	3	2
Best Intensity	1.83	**1.72**	2.06

TABLE I: Hyperparameters and Best Intensities for Different Figures

The Table I summarizes the optimal hyperparameters obtained for the Firefly Algorithm in different experimental setups. For each figure, the table presents the learning rate (lr), β_1, β_2, embedding dimension, window size (w), and the best intensity value achieved. These hyperparameters were fine-tuned to minimize the loss function, demonstrating the effectiveness of the Firefly Algorithm in hyperparameter optimization.

B. *Training CBoW with Best Hyperparameters*

After using the Firefly Algorithm for hyperparameter tuning, we determined the best hyperparameters, including the learning rate, β_1, β_2, etc. The training loss plot in Figure 6 illustrates the optimization process, showing how the loss decreases over training iterations.

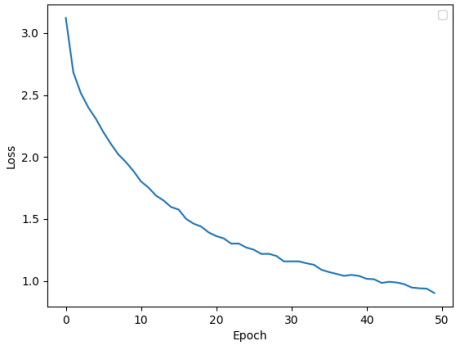

Fig. 6: Training Loss Plot

Following training, we extracted the embedding vectors for each word, such as 'king' and 'queen'. In this experiment, we used a 4-dimensional embedding dimension to represent

each word's context. We applied Principal Component Analysis (PCA) for dimensionality reduction, visualizing the embeddings in a 2D space for better understanding.

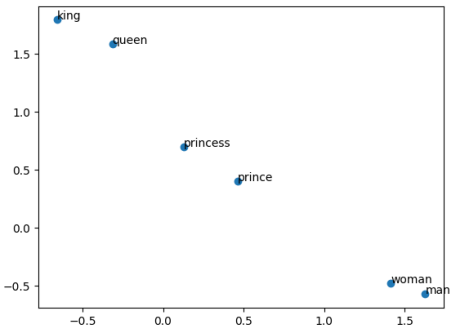

Fig. 7: Word Representation in 2D Space

Figure 7 visualizes the word representations in a 2D space, where each point represents a word. We can observe that the word 'king' is close to 'queen', indicating that their representations are similar and capture their semantic relationship. Similarly, 'man' and 'woman' are also close in the representation space, showing that the model has learned meaningful relationships between words.

IV. Conclusion

The Firefly Algorithm effectively optimized hyperparameters for the CBoW model, leading to meaningful word embeddings. The model learned semantic relationships, as seen in the 2D visualization. These results demonstrate the algorithm's efficacy in NLP tasks, such as sentiment analysis and text classification.

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[2] X.-S. Yang, "Firefly algorithms for multimodal optimization," 2010.