Implementation of Backpropagation and Training a Palindrome Network (Re-evaluation)

Waisullah Yousofi, 22d0382

19-04-2024

Problem Statement

- **Input**: 10-bit String (of numbers)
- **Output**: 1 if Palindrome, 0 otherwise

Architecture details(1/2)

- Due to its simplicity and efficiency in avoiding the vanishing gradient problem **ReLU (Rectified Linear Unit)** is used in the **hidden layer**.
- As we are dealing with binary classification problems **Sigmoid function** is used in the **output layer**.
- **CE** loss function is used to estimate the model error.

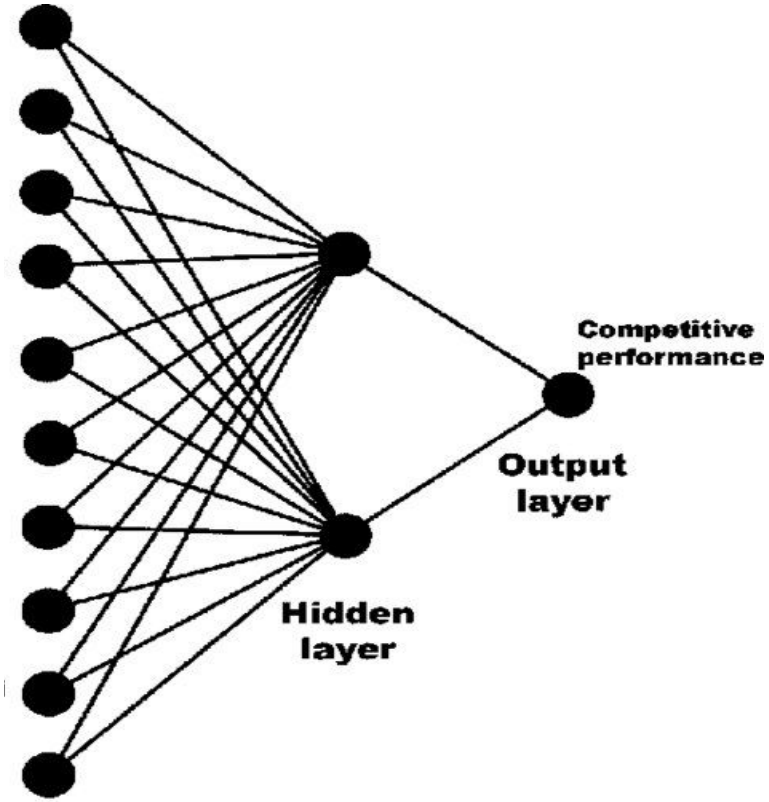

Figure-1-Ref

Hyperparameter details (1/2)

Learning Rate = 0.001

The learning rate controls how much the model's weights are updated during training. A smaller learning rate might require more training epochs but can lead to better convergence.

Number of Epochs = 20000

This is the number of times the entire training dataset is passed forward and backward through the neural network.

Hyperparameter details (2/2)

Weight Initialization = Xavier initialization

It is used to initialize the weights such that the variance of the activations are the same across every layer.

Cross-validation Strategy = 5 fold

5-fold cross-validation is used, which is a method of splitting the entire dataset into 5 different subsets (or folds) and using one fold for validation while the others for training. This repeats for each fold and helps in understanding the model's performance on different segments of the data.

Overall performance

- Overall Accuracy: 99.5
  - Class 0 Accuracy: 99.5
  - Class 1 Accuracy: 100.0

Interpretability of middle layer (1/2)

- **What are the hidden layer neurons computing?**
  - Has the job of activating neurons (**passing the weighted sum values**) of the hidden layer in response to input values.
  - If so, there must be patterns of neuron activations in response to a batch of inputs to the model?

- Technique we used to capture neuron firing patterns is:
  - NMF (Non-negative Matrix Factorization)

Interpretability of middle layer (2/2)

- **NMF (Nonnegative Matrix Factorization)**

Visualization on VSCode

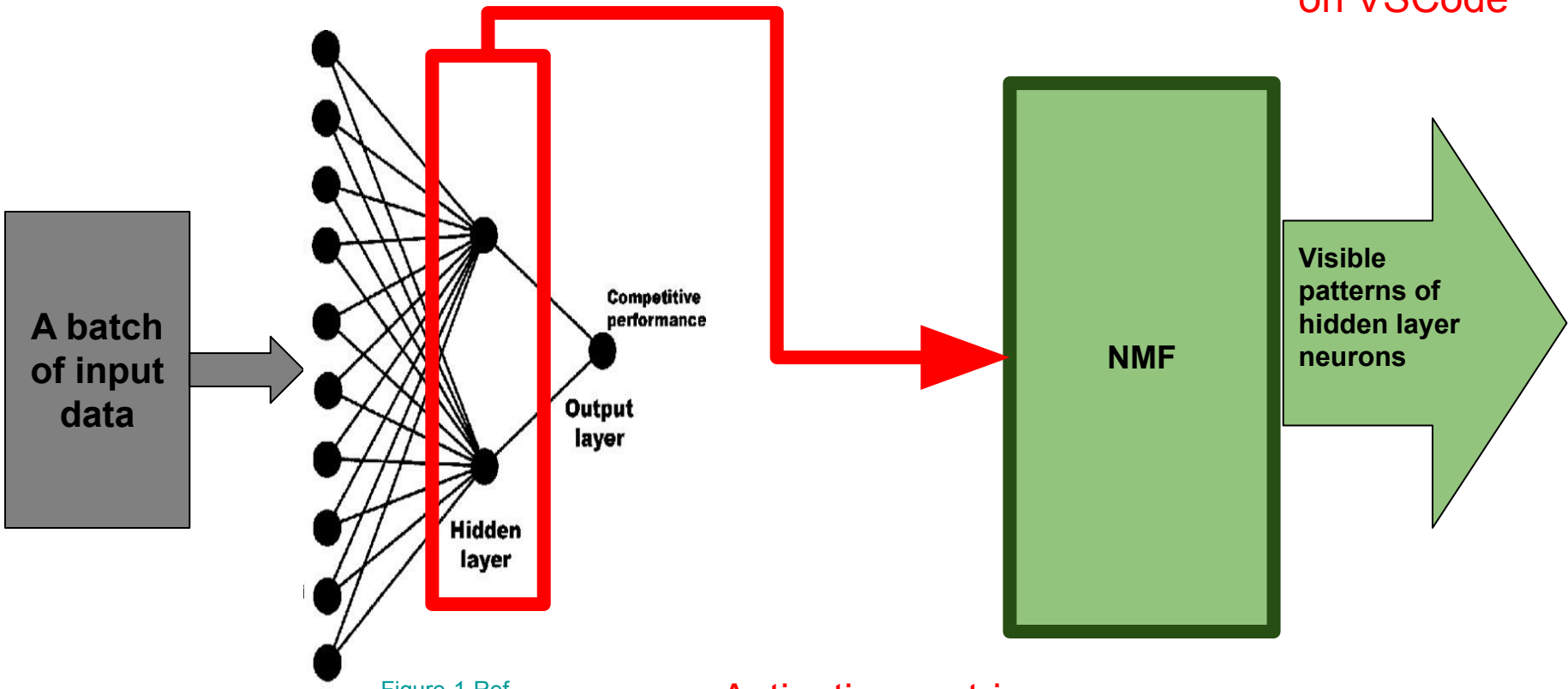

Figure-1-Ref

Activation matrix as input for NMF

Learnings(1/3)

Architecture Mastery

- **Layer Logic:** *Understood the roles of input, hidden, and output layers.*
- **Activation Functions:** *Discovered the potential of ReLU for hidden layers and sigmoid for binary outputs.*

Hyperparameter Tuning

- **Tuning Skills:** *Learned the art of balancing learning rate, epoch count, and neuron numbers.*
- **Optimization Quest:** *Discovered the importance of experimenting to find the best hyperparameter mix.*

Learnings(2/3)

The Importance of Data Preprocessing

- **Data Balancing Act:** *Gained proficiency in creating well-represented and balanced datasets.*

Model Evaluation

- *Evaluation of model using accuracy metric.*
- *Learned to check class-wise accuracy*

Learnings(3/3)

- Most importantly, we don't need ML to solve this particular problem as the feature size is fixed. **Even a PTA can do this job!**

- During training, model learns to construct a mirror matrix in the **input to hidden** layer matrix. i.e

  [[1,2]

  [-1,-2]]

Demo

Google collab code link:
Palindrom_Detection_Assignment 1_Updated_by_Waisullah.ipynb

Evaluation Scheme

- Correct implementation of BP from scratch: 10 marks (show the code parts that implement weight change rules)
- Theory of BP clarity: 10
- Overall Performance: accuracy >=90: 10 marks; 80-89: 9; 70-79: 8; 60-69: 7; 50-59: 40-49: 5; 30-39: 4. And so on
- Interpretability of middle layer: 10
- Demo -10