REPORT

Binary Emotion Classification

Atharva Bhawalkar

ABSTRACT

This project aims to predict the emotional state of *valence* (high or low) using EEG data from the DEAP[1] dataset. Primarily, valence and arousal are two dimensions used to describe emotional states. Valence refers to how positive or negative an emotion is. We have tried to implement a deep neural network for our aim of binary classifying valence state. We have fundamentally implemented the techniques for calculating the cost, gradient descent, forward and backward propagation.

INTRODUCTION TO THE DATASET

The electroencephalogram (EEG) and peripheral physiological signals of 32 participants were recorded as each watched 40 one-minute long excerpts of music videos. Participants rated each video in terms of the levels of arousal, valence, like/dislike, dominance and familiarity.

We decided to proceed with the preprocessed version of the data wherein:

The data was downsampled to 128Hz.

1. EOG artifacts were removed.
2. A bandpass frequency filter from 4.0-45.0Hz was applied.
3. The data was averaged to the common reference.
4. The data was segmented into 60 second trials and a 3 second pre-trial baseline removed.

Array name	Array shape	Array contents
data	40 x 40 x 8064	video/trial x channel x data
labels	40 x 4	video/trial x label (valence, arousal, dominance, liking)

Time Points: 8064 per trial

- **Sampling rate:** 128 Hz (128 samples per second)
- **Duration of each trial:** 63 seconds

Thus, the number of time points per trial for each channel is:

$$128 \text{ samples/second} \times 63 \text{ seconds} = 8{,}064 \text{ time points}$$

Hence, the total features = Time points x Number of channels = 8064 x 40 = 322560

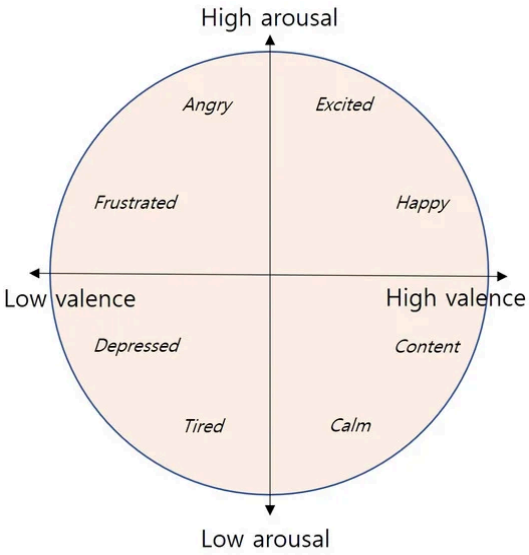
And, the total instances = Number of participants x Number of trials = 32 x 40 = 1280

[1] Link to the official dataset

METHODOLOGY

1. **Data Collection and Preprocessing:**

   - **Data Loading:**
     - Data is loaded from preprocessed `.dat` files using the `pickle` library.
     - A function `read_eeg_signal_from_file` is defined to handle the loading of each participant's data file.
   - **Data Structure:**
     - The data is organized into a 4D NumPy array with dimensions `(32, 40, 40, 8064)`, representing participants, trials, channels, and time points.
     - Labels for valence (emotional state) are stored in a separate array with dimensions `(32, 40, 4)`, representing the values of valence, arousal, dominance, and liking.



2. **Label Extraction and Binary Classification:**

   - Valence Labels: We extracted the first dimension of the labels array, which corresponds to valence. We restrict it to only valence in order to achieve the binary classification aim of the assignment.
   - Binary Classification: Convert the continuous valence labels into binary classes (0 or 1) based on a threshold of 5, where values greater than 5 are classified as high valence (1) and those less than or equal to 5 as low valence (0).

3. **Data Reshaping:**

   - Flattening Trials: The 4D data is reshaped into a 3D array by flattening each trial into a single feature vector, resulting in dimensions of `(32, 40, 322560)`.
   - Preparing for Training: The data is further flattened into a 2D array `(1280, 322560) for training, with the corresponding binary valence labels flattened into a 1D array.

4. **Train-Test Split:**

   Use `train_test_split` from `sklearn.model_selection` to divide the data into training and testing sets with an 80-20 split. The training set contains 1,024 samples, while the test set contains 256 samples.

5. **Data Normalization:**

   Standardization:The data is normalized using `StandardScaler` to standardize the features to have a mean of 0 and a variance of 1. This helps to improve the convergence of the neural network during training.

6. **Neural Network Design:**

   - Parameter Initialization: A function `initialize_parameters` to initialize weights and biases for the neural network layers using *He initialization* for weights and *zero* for biases.
   - Activation Functions:
     - Leaky ReLU: Used for the hidden layers to introduce non-linearity and avoid the dying ReLU problem.
     - Sigmoid: Used for the output layer to obtain binary classification probabilities.
   - Forward Propagation: Define the function `forward_propagation` to compute the activations of each layer and the output probabilities.
   - Cost function:
     - Binary Cross-Entropy: Implement the cost function `compute_cost` to calculate the loss between predicted and actual values.
   - Backward Propagation: The `backward_propagation` function to compute gradients of the weights and biases. This involves calculating the derivatives of the loss with

respect to each parameter, allowing the model to learn from errors.

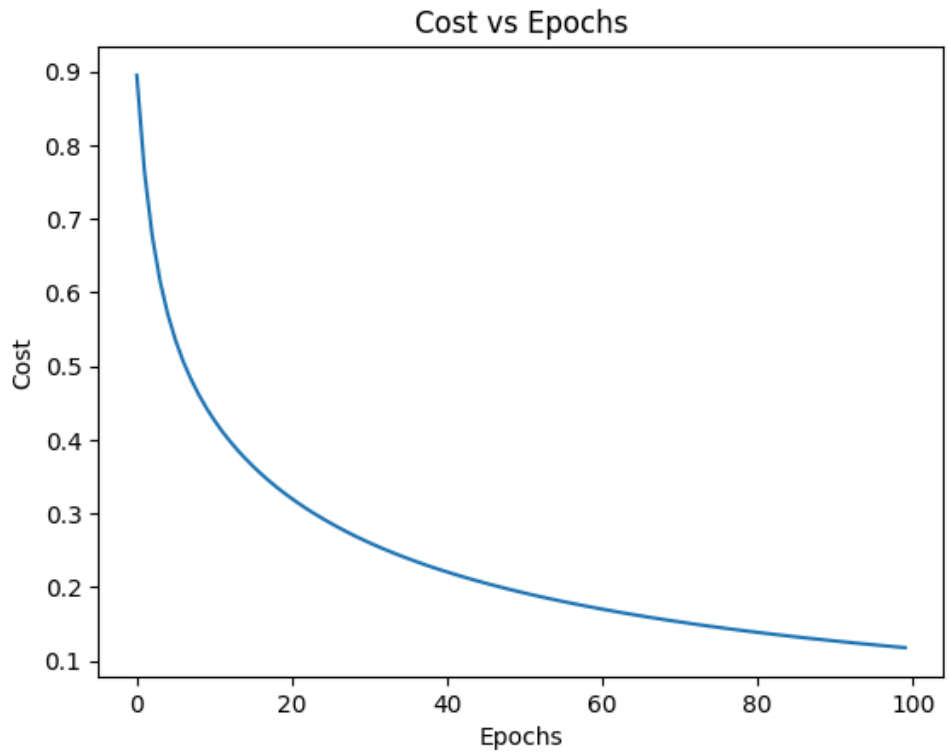
- Parameter Updates: The `update_parameters` function to update the weights and biases using the computed gradients and a specified learning rate.

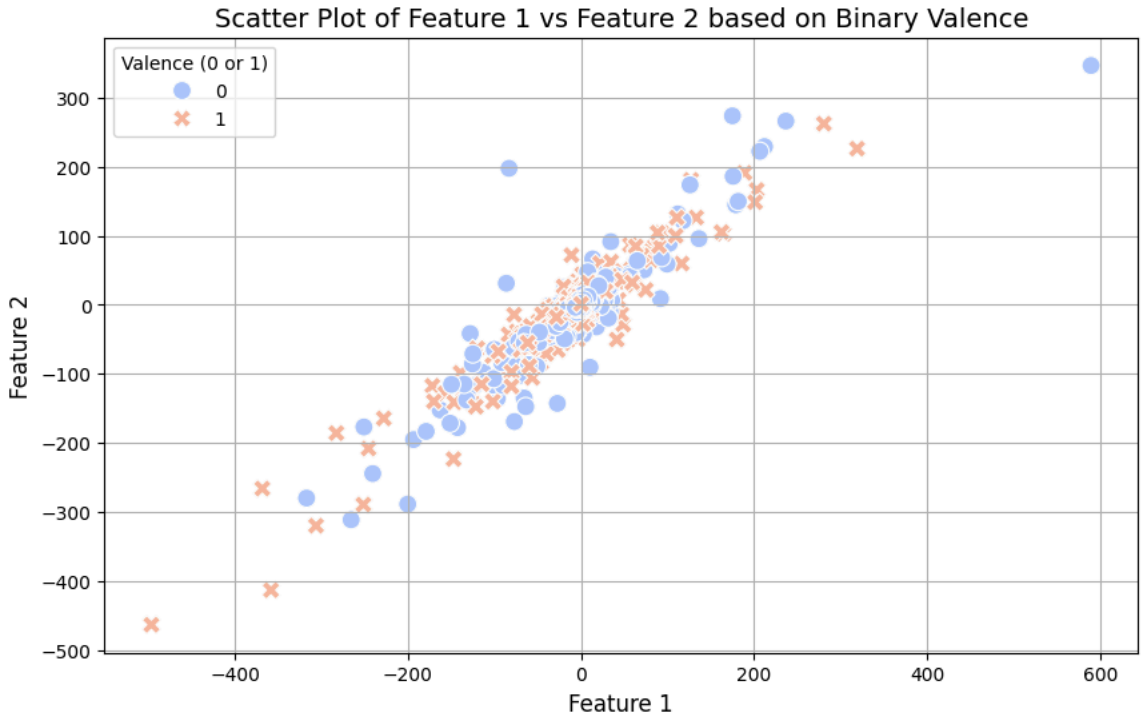
7. **Visualization:**

- Scatter Plots: Since there are many features, we can visualize the distribution of the first two features in a scatter plot based on the predicted binary valence labels.
- PCA Visualization: Principal Component Analysis (PCA) to reduce the dimensionality of the feature space for visualization.

RESULTS

The neural network achieved a training accuracy of **99.90%** and a testing accuracy of **60.55%**





OBSERVED PROBLEM TRENDS

1. **Curse of Dimensionality:** As the number of features increases, the complexity of the model can increase, potentially leading to overfitting. In high-dimensional spaces, data points can become sparse, and it becomes harder for the classifier to generalize.
2. **Relatively Small Sample Size:** DEAP contains recordings from only 32 participants, each watching 40 video clips, for a total of 1,280 trials. While the dimensionality is high (322,560 features), the number of trials is relatively small, which can lead to overfitting, especially with complex models like deep neural networks.