Contents:

Overview

This report outlines the architecture and design choices for an Artificial Neural Network (ANN) model developed for this project. The ANN model was chosen and tailored for binary classification tasks such as customer segmentation and churn prediction. The architecture has been optimised for performance based on training data properties and predictive accuracy. Below, we detail the configuration of the input, hidden, and output layers, as well as key design considerations such as activation functions, the loss function, and the optimiser used during training.

Model's Architecture

The ANN model is constructed as a feedforward neural network with multiple fully connected layers. It is designed to capture complex relationships in the input data and make accurate binary classification predictions. The architecture is structured to balance learning capacity, training stability, and computational efficiency.

Breakdown of the ANN model's layers:

1. Input Layer:
    - Purpose: The input layer is responsible for receiving the data points (features) fed into the model for processing.
    - Configuration: The input dimension is determined by the number of features in the dataset (e.g. `input_dim = number of columns in X_train.csv`).
    - Activation: No activation function is applied at this stage as it serves purely to pass data to the next layer.
2. Hidden Layers:
    - First Hidden Layer:
        - Number of Neurons: 64 neurons are chosen for this layer to provide sufficient capacity to capture initial complex patterns in the data.
        - Activation Function: ReLU (Rectified Linear Unit) is applied, offering several advantages:
            1. Mitigates the vanishing gradient problem by allowing gradients to propagate effectively through deep networks.
            2. Computationally efficient due to its simple max(0, x) operation.

        - Rationale: The relatively high number of neurons in this layer ensures that the network starts with a strong representation of the input features.
    - Second Hidden Layer:
        - Number of Neurons: 32 neurons, halving the size of the previous layer to gradually focus on more refined features.
        - Activation Function: ReLU continues to be used, allowing non-linear transformations and capturing intermediate data representations.
        - Purpose: This layer helps reduce dimensionality while maintaining sufficient representation power for learning key patterns.
    - Third Hidden Layer:
        - Number of Neurons: 16 neurons to further refine the data and extract the most meaningful patterns learned from prior layers.
        - Activation Function: ReLU, consistent with the earlier layers, ensures continuity in non-linear transformations.
        - Rationale: This layer acts as a bridge, condensing complex information into a smaller set of highly relevant features before passing it to the output.
3. Output Layer:
    - Number of Neurons: 1 neuron(as this is a binary classification task).
    - Activation Function: Sigmoid, which maps the output to a probability range between 0 and 1. This is essential for binary classification, as it allows easy interpretation of output probabilities and thresholding (e.g., > 0.5 as class 1, ≤ 0.5 as class 0).
    - Purpose: The final prediction of the model, which can be used for decision-making or further analysis.

Activation Functions

- ReLU (Rectified Linear Unit):
    - Advantages:
        - Helps the network learn faster by avoiding the vanishing gradient problem.
        - Computationally simple, making training more efficient.
    - Drawbacks:
        - Can suffer from "dead neurons" when weights are updated such that the output is consistently zero. This can be mitigated with variants like Leaky ReLU if needed.
- Sigmoid Activation:
    - Role in Output Layer:

        - Transforms the raw output to a value between 0 and 1, representing a probability.
    - Pros and Cons:
        - Pro: Ideal for binary classification due to its probabilistic interpretation.
        - Con: Can cause vanishing gradients in deep networks when used in hidden layers, but suitable in the output.

Loss Function and Optimization Strategy

- Loss Function:
    - Binary Cross-Entropy Loss is used. It is suitable for comparing predicted probabilities to the true binary class labels, minimizing the discrepancy between the two.
- Optimizer:
    - Adam Optimizer is chosen due to its adaptive learning rate capability. This helps balance training speed and stability, adjusting the learning rate for each parameter dynamically based on gradient history.
    - Initial Learning Rate: Set at 0.0005, which provides a good starting point for many tasks. Adjustments may be made based on validation performance.

Training Configuration

- Batch Size:
    - Value: 64 samples per batch.
    - Reasoning: This batch size allows for efficient parallel processing on modern GPUs, leveraging their computational power and memory capacity for faster training. Batch size 64 offers a smoother gradient descent path due to more stable gradient estimates compared to smaller batch sizes like 32.
    - Trade-offs:
        - Training Speed: Using batch size 64 speeds up the training process per epoch as more samples are processed simultaneously.
        - Gradient Updates: Fewer gradient updates per epoch compared to batch size 32, resulting in smoother, more consistent gradient updates that can lead to faster initial convergence.
        - Generalisation: While larger batch sizes can sometimes reduce the noise in gradient updates, which may impact the ability of the model to generalize as well as smaller batch sizes, batch size 64 strikes a good balance between stability and generalization.

- Number of Epochs:
  - Set to 100 epochs for initial training. This value is chosen to allow sufficient iterations for convergence without overfitting. Early stopping mechanisms or validation checks can be introduced in future iterations.

Conclusion

This ANN model architecture is optimised for predictive performance, balancing complexity with efficiency. It can effectively learn patterns in training data and make robust predictions on unseen data, making it suitable for applications such as customer segmentation and churn prediction. With further enhancements, such as regularization techniques and hyperparameter tuning, the model's performance can be tailored to specific business needs and datasets.